

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Jonathan Paul Driemeyer Brown

**Chatronics: utilizando GPTs para auxílio a
projeto de sistemas de aquisição de sinais**

Porto Alegre

2024

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Jonathan Paul Driemeyer Brown

**Chatronics: utilizando GPTs para auxílio a projeto de
sistemas de aquisição de sinais**

Projeto de Diplomação II, apresentado ao Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do grau de Engenheiro Eletricista

UFRGS

Orientador: Prof. Dr. Tiago Oliveira Weber

Porto Alegre

2024

Jonathan Paul Driemeyer Brown

Chatronics: utilizando GPTs para auxílio a projeto de sistemas de aquisição de sinais

Projeto de Diplomação II, apresentado ao Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do grau de Engenheiro Eletricista

BANCA EXAMINADORA

Prof. Dr. Alexandre Balbinot
UFRGS

Prof. Dr. Tiago Roberto Balen
UFRGS

Prof. Dr. Tiago Oliveira Weber
Orientador - UFRGS

Agradecimentos

A experiência universitária é composta por muito mais que apenas aulas e provas. A conquista de conseguir se formar não é exclusivamente decorrente do quanto se estuda, mas também do apoio que se teve ao longo dos anos.

Agradeço, em especial, à minha mãe Debora e ao meu pai Tim, que sempre me instigaram a questionar o mundo ao meu redor e a nunca desistir dos meus sonhos, mesmo em momentos difíceis. Tive mais dificuldades em por em palavras o carinho que tenho por vocês do que escrever a monografia, mas, em resumo, amo vocês.

À minha Vó, America, aos meus dindos - Katia e Ricardo, aos meus tios - Jefferson e Grasi, aos meus primos, agradeço pelo forte núcleo familiar que firmamos, em que todos sempre estão apoiando uns aos outros.

À Luiza e à Mayara, por se tornarem tão próximas de mim, sempre dispostas a uma sessão de fofocas e conversas. Ao grupo do amigo secreto da elétrica, por trazerem vida ao corredores e as salas de aula da UFRGS.

Ao André, por sempre apoiar uma ida ao bar para se divertir e desopilar dos trâmites do cotidiano. Ao Bruno, por ter me aproximado da música, que foi muito importante para mim. E a ambos pelas idas a shows, mesmos que, às vezes, duvidosos.

À Gabi, por ter me ajudado a passar pelo último ano do curso com um sorriso no rosto, além de ter sido de tamanha ajuda na revisão do trabalho.

Aos meus colegas de trabalho na SAP, e ao meu gerente, Cláudio, que me ajudaram a me encontrar na área de engenharia de *software*, além de proporcionarem um ambiente bom de trabalhar e se desenvolver.

Por fim, agradeço ao Tiago por me orientar desde a iniciação científica em 2021, que, além de me motivar a adentrar na área de inteligência computacional, foi importante para o meu desenvolvimento como acadêmico e profissional.

Resumo

Modelos especialistas têm sido um dos focos da academia e da indústria, uma vez que procuram cumprir tarefas atribuídas a especialistas da área, assim otimizando processos e custos. Contudo, os grandes modelos de linguagem pré-treinados permitem que ferramentas especializadas possam ser desenvolvidas sem exigir o próprio *dataset* ou poder computacional para treinamento, uma vez que a engenharia de *prompt* tem se mostrado promissora para diversas aplicações. Os últimos meses têm sido caracterizados pelo crescimento no número de pesquisas que utilizam *Generative Pre-Trained Transformers* (GPTs), em que os autores procuram encontrar novas aplicações para os modelos conversacionais - os *chat-bots*, mesmo que estes não sejam inseridos na área de linguísticas. Entretanto, não há na literatura a utilização de modelos de linguagem focados em auxílio a projeto de sistemas de aquisição de sinal. Portanto, este trabalho apresenta uma ferramenta com o intuito de fornecer sugestões de sistemas de aquisição de sinal seguindo uma metodologia de síntese a nível de sistema por uma abordagem *top-down*. Dessa maneira, por meio do *GPT-4-Turbo*, o aplicativo fornece o diagrama da arquitetura do projeto e, em seguida, apresenta os detalhes de implementação de cada bloco. No total, a ferramenta dispõe de 26 *prompts* distintos, nos quais as técnicas de personas, perguntas e *templates* foram empregadas. Para testar a ferramenta, se desenvolveu um método novo de teste, no qual dois GPTs adicionais são utilizados, um para emular o usuário humano e outro para avaliar a qualidade da solução. Com o intuito de aproximar os testes do caso de uso proposto, foram selecionados dois projetos da disciplina de Instrumentação A da UFRGS e dois projetos do livro *Data Acquisition for Sensor Systems* de Taylor (1997) para compor o conjunto de *testbenches*. Além de testar o desempenho do modelo em gerar sugestões adequadas, foram implementados dois cenários distintos de contexto, uma nas quais todos as restrições de projeto são fornecidas diretamente ao modelo, e outra na qual o modelo é obrigado a fazer perguntas ao usuário para obtê-las. Assim, após 160 iterações de teste, o trabalho concluiu que existe um potencial das LLMs se tornarem adequados para a síntese e/ou auxílio ao projeto de sistemas de aquisição de sinal, mas que ainda existem limitações tecnológicas. Os resultados apresentam arquiteturas e topologias coerentes com o solicitado, porém os GPTs têm dificuldades em considerar simultaneamente todos os requisitos e, ocasionalmente, cometem erros teóricos. A hipótese de que os diferentes contextos - com ou sem necessidade de perguntas - são igualmente efetivos foi rejeitada para metade dos *testbenches*.

Palavras-chave: GPT; LLM; Sistemas de Aquisição de Sinais; Síntese a Nível de Sistema.

Abstract

Expert models have been a focus in the industry and in academia, given they aim to fulfil tasks attributed to specialists in the field, therefore optimizing processes and costs. However, pre-trained large language models have enabled the development of specialized tools without the need for much computational power, or their own dataset for training, given that prompt engineering has shown to be promising in many fields. In recent months, there has been an increase in the number of projects using Generative Pre-Trained Transformers (GPTs), where the authors are aiming to discover new applications for large language models, even beyond the field of linguistics. However, no research on language models being used for the design of data acquisition systems has been found in literature, therefore, the aim of this study is to introduce the development of an application that provides a comprehensive design for data acquisition systems using a system-level synthesis methodology with a top-down approach. Utilizing GPT-4-Turbo, the tool offers the user a diagram of the suggested architecture and later provides implementation details for each block. In total, the tool has 26 distinct prompts at its disposal, in which persona, questions and template techniques were used. To run automated tests, a novel method was developed, employing two additional GPTs — one for emulating the human user and another for evaluating the quality of the final solution. In order to close the tests for the proposed user case, two projects from the Instrumentation A laboratories in UFRGS and two projects from the book, Taylor (1997), were selected to compose the testbenches. Besides testing the tools capabilities for generating adequate suggestions, two distinct types of contexts were implemented; one of which has access to all project requirements, whilst the other is forced to ask questions to get them. After 160 test iterations, the study concludes that there is potential for these models to serve adequately as synthesis/assistant tools for data acquisition systems, but there are still technological limitations. The results show coherent architectures and topologies, but that GPTs have difficulties in simultaneously considering all requirements and occasionally commit theoretical errors. The hypothesis that the different contexts are equally effective - with or without the need for questions - was rejected for half of the testbenches.

Keywords: GPT; LLM; Data Acquisition Systems; System Level Synthesis.

Lista de Figuras

Figura 1 – Arquitetura proposta do Transformador.	22
Figura 2 – Diagrama de blocos representando o cálculo do vetor de atenção.	23
Figura 3 – Divisão de texto em unidades de <i>tokens</i>	25
Figura 4 – Arquitetura do GPT original.	26
Figura 5 – Comparação entre os métodos <i>in-context</i> com o <i>fine-tuning</i>	28
Figura 6 – Diferentes abordagens da síntese a nível de sistema.	41
Figura 7 – Pirâmide de teste.	42
Figura 8 – Fluxo da metodologia experimental.	47
Figura 9 – Diagrama de blocos da visão geral do fluxo de conversa proposto.	51
Figura 10 – Fluxo de conversa da etapa de arquitetura.	52
Figura 11 – Fluxo de conversa da etapa de detalhamento.	55
Figura 12 – Uma das possíveis arquiteturas para o projeto de medição de posição angular.	66
Figura 13 – Possível solução do laboratório de termometria de Instrumentação A	68
Figura 14 – Arquitetura sugerida para o projeto de um medidor de vibração portátil.	71
Figura 15 – Arquitetura sugerida do exemplo do livro <i>Data Acquisition Systems</i>	73
Figura 16 – Fluxo de testes utilizados para avaliar a ferramenta e os testes automatizados.	76
Figura 17 – Diagrama do melhor resultado para o laboratório de medição de posição angular, obtida da iteração 7 com contexto direto.	85
Figura 18 – Diagrama do melhor resultado para o laboratório de termometria, obtida da iteração 3 com contexto investigativo.	86
Figura 19 – Diagrama do melhor resultado para o projeto de acelerometria, obtida da iteração 18 com contexto direto.	88
Figura 20 – Diagrama do melhor resultado para o projeto de medição de pressão e temperatura superficial, obtida da iteração 8 com contexto direto.	89
Figura 21 – Diagrama de blocos usual do laboratório de medição da posição angular obtida da iteração 12 com contexto investigativo.	101
Figura 22 – Diagrama de blocos do laboratório de medição da posição angular obtida da iteração 10 com contexto direto, em que o fluxo proposto está incorreto e confuso.	102
Figura 23 – Diagrama de blocos do laboratório de medição da posição angular obtida da iteração 20 com contexto direto, em que incorretamente está indicada tensão diferencial.	102

Figura 24 – Diagrama de blocos do laboratório de termometria obtida da iteração 10 com contexto direto, em que incorretamente estão indicadas 3 diferentes etapas de amplificação.	103
Figura 25 – Diagrama de blocos correto do laboratório de termometria obtida da iteração 3 com contexto investigativo.	103
Figura 26 – Diagrama de blocos do laboratório de acelerometria obtida da iteração 6 com contexto direto, em que um passa-altas é empregado.	104
Figura 27 – Diagrama de blocos do laboratório de acelerometria obtida da iteração 10 com contexto investigativo, em que a frequência de 0,25 Hz não é atenuada.	104
Figura 28 – Diagrama de blocos do laboratório de acelerometria obtida da iteração 4 com contexto investigativo, em que apenas um passa-baixas são empregados.	104
Figura 29 – Diagrama de blocos do laboratório de pressão e temperatura superficial retirado de um teste preliminar, em que os arranjos de sensores estão expandidos.	106
Figura 30 – Diagrama de blocos do laboratório de pressão e temperatura superficial da iteração 7 com contexto investigativo, em que os sensores estão em arranjos.	106
Figura 31 – Diagrama de blocos do laboratório de pressão e temperatura superficial da iteração 16 com contexto direto, em que a apresentação ficou parcialmente expandida.	107
Figura 32 – Diagrama de blocos do laboratório de pressão e temperatura superficial da iteração 12 com contexto direto, em que os MUXs estão corretamente indicados.	107

Lista de Tabelas

Tabela 1 – Comparação das funcionalidades dos modelos GPT, a partir do GPT-3.5-Turbo.	30
Tabela 2 – Precificação dos modelos GPT-3.5 e GPT-4.	30
Tabela 3 – Ferramentas computacionais utilizadas.	49
Tabela 4 – Comparação entre os contextos direto e investigativo.	58
Tabela 5 – Comparação entre os <i>testbenches</i>	65
Tabela 6 – Resultados das simulações do <i>testbench</i> do laboratório de medição de posição angular com contexto direto.	91
Tabela 7 – Resultados das simulações do <i>testbench</i> do laboratório de medição de posição angular com contexto informativo.	92
Tabela 8 – Dados calculados a partir dos resultados do <i>testbench</i> de medição de posição angular	92
Tabela 9 – Resultados das simulações do <i>testbench</i> do laboratório de termometria com contexto direto.	93
Tabela 10 – Resultados das simulações do <i>testbench</i> do laboratório de termometria com contexto informativo.	94
Tabela 11 – Dados calculados a partir dos resultados do <i>testbench</i> de termometria .	94
Tabela 12 – Resultados das simulações do <i>testbench</i> do projeto de acelerometria com contexto direto.	95
Tabela 13 – Resultados das simulações do <i>testbench</i> do laboratório de acelerometria com contexto informativo.	96
Tabela 14 – Dados calculados a partir dos resultados do <i>testbench</i> de acelerometria.	96
Tabela 15 – Resultados das simulações do <i>testbench</i> do projeto de pressão e temperatura superficial com contexto direto.	97
Tabela 16 – Resultados das simulações do <i>testbench</i> do laboratório de pressão e temperatura superficial com contexto informativo.	98
Tabela 17 – Dados calculados a partir dos resultados do <i>testbench</i> de pressão e temperatura superficial	98
Tabela 18 – Valores do U-Test para o <i>testbench</i> de medição de posição angular. . .	109
Tabela 19 – Valores do U-Test para o <i>testbench</i> de termometria.	109
Tabela 20 – Valores do U-Test para o <i>testbench</i> de acelerometria.	109
Tabela 21 – Valores do U-Test para o <i>testbench</i> de pressão e temperatura superficial.	110
Tabela 22 – Resultados dos testes de significância.	110
Tabela 23 – Valores críticos de U para $\alpha = 0,05$	149

Lista de abreviaturas

ADC	Conversor Analógico Digital (do inglês, Analog-to-Digital Converter)
ALPAC	Comitê Consultivo de Processamento Automático de Linguagem (do inglês, Automatic Language Processing Advisory Committee)
AMPOP	Amplificador Operacional
BTPP	Retropropagação no Tempo (do inglês, Backpropagation Through Time)
CNN	Rede Neural Convolutacional (do inglês, Convolutional Neural Network)
CLI	Interface de Linha de Comando (do inglês, Command Line Interface)
CPU	Unidade Central de Processamento (do inglês, Central Processing Unit)
CSS	Folhas de Estilo em Cascata (do inglês, Cascading Style Sheets)
DAC	Conversor Digital Analógico (do inglês, Digital-to-Analog Converter)
FET	Transistor de Efeito de Campo (do inglês, Field Effect Transistor)
FNN	Rede Neural Feedforward (do inglês, Feedforward Neural Network)
GPT	Transformador Pré-treinado Generativo (do inglês, Generative Pre-trained Transformer)
GNU	GNU's not UNIX
GRU	Unidade Recorrente Fechada (do inglês, Gated Recurrent Unit)
GPU	Unidade de Processamento Gráfico (do inglês, Graphics Processing Unit)
HTML	Linguagem de Marcação de Hipertexto (do inglês, HyperText Markup Language)
HTTP	Protocolo de Transferência de Hipertexto (do inglês, HyperText Transfer Protocol)
IA	Inteligência Artificial
JSON	Notação de Objetos JavaScript (do inglês, JavaScript Object Notation)
LSTM	Memória de Curto Longo Prazo (do inglês, Long Short-Term Memory)
LLM	Grande Modelo de Linguagem (do inglês, Large Language Model)
MIT	Instituto de Tecnologia de Massachusetts (do inglês, Massachusetts Institute of Technology)
MLP	Perceptron de Múltiplas Camadas (do inglês, Multi-Layer Perceptron)

MT	Tradução de Máquina (do inglês, Machine Translation)
MUX	Multiplexador
NLP	Processamento de Linguagem Natural (do inglês, Natural Language Processing)
NTC	Coefficiente de Temperatura Negativo (do inglês, Negative Temperature Coefficient)
PNG	Gráficos de Rede Portáteis (do inglês, Portable Network Graphics)
RAM	Memória de Acesso Aleatório (do inglês, Random Access Memory)
RTD	Detector de Temperatura de Resistência (do inglês, Resistance Temperature Detector)
RNN	Rede Neural Recorrente (do inglês, Recurrent Neural Network)
SVG	Gráficos Vetoriais Escaláveis (do inglês, Scalable Vector Graphics)
TCC	Trabalho de Conclusão de Curso
TTS	Texto para Fala (do inglês, Text-to-Speech)
UFRGS	Universidade Federal do Rio Grande do Sul
UI	Interface do Usuário (do inglês, User Interface)
USD	Dólar dos Estados Unidos (do inglês, United States Dollar)

Sumário

1	INTRODUÇÃO	14
1.1	Considerações Iniciais	14
1.2	Justificativa	15
1.3	Objetivos	16
1.3.1	Objetivo Geral	16
1.3.2	Objetivos Específicos	16
1.4	Contribuições do Trabalho	16
1.5	Considerações Éticas	17
1.6	Estrutura do Trabalho	18
2	FUNDAMENTAÇÃO TEÓRICA	19
2.1	Modelos Transformadores	19
2.1.1	Contexto Histórico	19
2.1.2	A Arquitetura e as Vantagens dos Transformadores	20
2.1.3	Transformadores Pré-treinados Generativos	24
2.1.3.1	Arquitetura	24
2.1.3.2	Técnicas de Treinamento	27
2.1.3.3	Modelos Disponíveis	29
2.1.3.4	API	31
2.1.4	Engenharia de <i>Prompt</i>	35
2.1.4.1	Personas	35
2.1.4.2	Perguntas	36
2.1.4.3	<i>Templates</i>	37
2.2	Sistemas de Aquisição de Sinais	38
2.2.1	Fluxo Clássico de Projeto	38
2.2.2	Síntese a Nível de Sistema	40
2.3	Testes Unitários e a Pirâmide de Testes	42
2.4	<i>Mann–Whitney U-Test</i>	43
2.5	Estado da Arte	45
2.5.1	Projetos Baseadas em LLMs	45
2.5.2	Ferramentas pra Síntese de Projetos	46
2.5.3	Trabalhos que Implementam LLM na Área de Engenharia Elétrica	46
3	METODOLOGIA	47
3.1	Materiais e Ferramentas	48
3.1.1	<i>Hardware</i> Utilizado	48

3.1.2	Software Utilizado	48
3.1.2.1	Modelos de Linguagem	48
3.1.2.2	Linguagem de Desenvolvimento	49
3.2	Engenharia de <i>Prompt</i>	50
3.3	Fluxo de Conversa	50
3.3.1	Visão Geral	50
3.3.2	Desenvolvimento da Arquitetura	52
3.3.3	Categorização dos Blocos	53
3.3.4	Detalhamento dos Blocos	55
3.3.5	Revisão	56
3.4	Testes	57
3.4.1	Execuções Automatizadas	57
3.4.1.1	Modelo Avaliador	59
3.4.2	Testes Unitários	63
3.5	Testbenches	64
3.5.1	Laboratório de Medição de Posição Angular	65
3.5.2	Laboratório de Termometria	67
3.5.3	Acelerômetro Portátil para Vibrações de Baixa Frequência	69
3.5.4	Monitor de Pressão e Temperatura Superficial de Maquinário	73
3.6	Análise dos Resultados	75
3.6.1	Visão geral	75
3.6.2	Execuções dos Testes Automatizados	76
3.6.3	Validação Manual	77
3.6.4	Análise e Exposição dos Dados	77
3.7	Aplicação CLI	78
3.7.1	Comandos	78
3.7.2	Experiência do Usuário	79
3.7.3	Logs e Histórico de Execuções	80
4	RESULTADOS	82
4.1	Funcionamento da Ferramenta	82
4.1.1	Testes Unitários	82
4.1.2	Erros Obtidos	83
4.2	Melhores Resultados por <i>Testbench</i>	83
4.2.1	Laboratório de Medição de Posição Angular	84
4.2.2	Laboratório de Termometria	84
4.2.3	Projeto de Acelerometria	87
4.2.4	Projeto de Pressão e Temperatura Superficial	87
4.3	Resultados de Testes com Notas e Vereditos do Modelo Avaliador .	90
4.3.1	Laboratório de Medição de Posição Angular	90

4.3.2	Laboratório de Termometria	93
4.3.3	Projeto de Acelerometria	95
4.3.4	Projeto de Pressão e Temperatura Superficial	97
4.4	Validação manual	99
4.4.1	Laboratório de Medição de Posição Angular	99
4.4.2	Laboratório de Termometria	102
4.4.3	Projeto de Acelerometria	103
4.4.4	Projeto de Pressão e Temperatura Superficial	105
4.4.5	Discussão sobre o Desempenho do Modelo Avaliador	107
4.5	Comparação dos Contextos	108
4.5.1	<i>Mann-Whitney U-Test</i>	108
4.6	Custo e Consumo de <i>Tokens</i>	111
5	CONCLUSÕES	112
	REFERÊNCIAS BIBLIOGRÁFICAS	114
	APÊNDICE A – PROMPTS	119
A.1	Arquitetura	119
A.2	Categorização	120
A.3	Detalhamento	121
A.4	Revisão	130
	APÊNDICE B – MELHORES RESULTADOS	132
B.1	Laboratório de Medição de Posição Angular	132
B.2	Laboratório de Termometria	135
B.3	Projeto de Acelerometria	139
B.4	Projeto de Pressão e Temperatura Superficial	143
	APÊNDICE C – TABELA PADRÃO DE U	149

1 Introdução

1.1 Considerações Iniciais

O número anual de publicações na área de inteligência artificial (IA) vem crescendo consistentemente na última década. Em 2010 cerca de 200 mil trabalhos foram publicados, um número considerável por si só; no entanto, em 2022 essa quantidade quase dobrou, ficando próxima de atingir a marca dos 500 mil (CSET, 2022). Contudo, esses dados ainda não contabilizam o pico súbito de interesse por IA generativa e, mais especificamente, por *chat-bots*.

Com o lançamento do ChatGPT, em 30 de Novembro de 2022 (OPENAI, 2022), o mundo percebeu que tecnologias baseadas em inteligência artificial não precisavam mais ser elitizadas para grupos de pesquisa - qualquer pessoa podia simplesmente acessar o site da OpenAI e utilizar o estado-da-arte em processamento de linguagem natural (NLP, do inglês, *Natural Language Processing*). O impacto foi tal que esse foi o assunto mais pesquisado na Wikipédia em 2023 (Mekouar, 2023), levando o site do ChatGPT a bater o recorde de aplicativo mais rápido a atingir 100 milhões de usuários, passando de redes como *Facebook* e *Instagram* (Hu, 2003).

Um dos principais motivos para o aumento drástico de pesquisas e ferramentas que utilizam transformadores pré-treinados generativos ¹ é a facilidade de implementação. Com seu uso, não se faz necessário considerar etapas de modelagem, treinamento, validação, e todos os outros passos envolvidos no desenvolvimento de um arquitetura inteligente, já que as APIs expostas pela OpenAI permitem o uso de qualquer um dos modelos da organização, sejam eles de texto, imagem, voz, entre outros, sem que seja preciso realizar todas essas etapas. Portanto, com algumas chamadas HTTP, pode-se usufruir dos maiores modelos existentes, que possuem escalas impossíveis de serem individualmente construídas.

Dessa forma, por meio da versão mais recente dos GPTs - o *gpt-4-1106-preview* ², neste trabalho desenvolveu-se uma ferramenta capaz de fornecer detalhes de implementação da síntese a nível de sistema, para projetos de sistemas de aquisição de sinais. Por meio de uma abordagem *top-down*, o usuário é solicitado a providenciar um enunciado inicial, que contextualize o projeto que se deseja sintetizar. Por meio de uma série de etapas de perguntas e respostas, a ferramenta busca delimitar mais o escopo do projeto e as preferências do usuário, que pode, por exemplo, utilizar um modelo específico de

¹ O termo em inglês, *Generative Pre-trained Transformer* (GPT), é utilizado para descrever a família de modelos transformadores desenvolvido pela OpenAI.

² Chamado também de *GPT-4-Turbo*, este modelo foi anunciado em 11 de novembro de 2023 (OPENAI, 2023c), tornando-se o mais recente da família dos GPTs até o momento da apresentação deste trabalho.

amplificador operacional (AMPOP). Após todas as perguntas serem respondidas, e o modelo gerar todo o detalhamento da implementação, um resumo é gerado de modo a compilar todas as informações necessárias para que o usuário possa seguir com a síntese.

Uma vez que não há na literatura o desenvolvimento de ferramentas similares, foi necessário implementar um método inédito de teste, capaz de emular a interação usuário-modelo. Dessa forma, um GPT adicional foi configurado para responder perguntas relacionadas aos requisitos do sistema de aquisição de sinal, tal qual um usuário humano. Além disso, ao terceiro modelo foi atribuída a função de avaliar a qualidade da solução proposta. Por conseguinte, o presente trabalho conta com três grandes modelos de linguagem (LLM, do inglês, *Large Language Model*).

No entanto, nem sempre é necessário emular o usuário, pois pode-se simplesmente querer providenciar um enunciado inicial contendo todos os detalhes mandatórios para a implementação, e não responder nenhuma pergunta subsequente. Para validar qual abordagem é melhor (com ou sem perguntas) escolheu-se o método estatístico *Mann-Whitney U-Test* para testar a hipótese de que não há diferença significativa entre estratégias.

No total, foram executadas 160 execuções da ferramenta, provenientes de 4 *testbences* selecionados para validarem a qualidade da solução fornecida. Dois desses projetos foram adaptados dos laboratórios de Instrumentação A da UFRGS, baseados em Balbinot e Brusamarello (2010), enquanto os outros dois foram extraídos de Taylor (1997). Os resultados indicam que existe potencial dos GPTs serem utilizados para essa aplicação, mas que ainda não estão prontos para tal. Após manualmente validar as soluções, observaram-se casos de alucinação³, além de más escolhas de projetos, que não necessariamente estão erradas, apenas indicam qualidade reduzida.

1.2 Justificativa

Com a disponibilização dos transformadores generativos pré-treinados por APIs, tornou-se possível estar apenas a uma chamada HTTP de distância de usufruir do estado da arte em inteligência artificial. Assim, o desafio passa a ser encontrar possíveis aplicações às LLMs, em quaisquer áreas que se possa imaginar. O principal motivador desta pesquisa se dá em razão de uma lacuna observada na interface de sistemas aquisição de sinais com modelos conversacionais, de modo a despertar a curiosidade de descobrir o potencial que existe nesse estudo.

Não é esperado que a ferramenta desenvolvida esteja pronta para uso real. No entanto, entende-se que a pesquisa desenvolvida deve incentivar incrementos a essa linha

³ Termo utilizado para descrever a situação em que a LLM providencia respostas erradas como se fossem verdades.

de estudo para que, em um futuro breve, aplicativos como esse tornem-se cotidianos no dia a dia do profissional e do estudante.

Deseja-se, também, aproveitar a oportunidade para aproximar a engenharia elétrica da engenharia de software. O autor deste estudo acredita que existe um grande potencial de crescimento para as engenharias, caso adotem práticas da ciência da computação. Historicamente, a área de engenharia elétrica apresenta resistência à mudanças, como destaca um dos criadores do Spectre⁴, ao dizer engenheiros desenvolvem uma espécie de síndrome de Estocolmo após se acostumarem a usar ferramentas difíceis e datadas (Kundert, 2011). Atualmente, a escrita de código é praticamente uma exigência para se realizar pesquisa de ponta, portanto, boas práticas de código somente podem trazer benefícios à profissão.

1.3 Objetivos

1.3.1 Objetivo Geral

Desenvolver uma ferramenta utilizando o GPT-4-Turbo capaz de auxiliar o projeto de sistemas de aquisição de sinais a nível de sistema, adotando uma abordagem *top-down* dirigida por restrições.

1.3.2 Objetivos Específicos

- Compor um conjunto de *testbenches*, formulados a partir de projetos reais de instrumentação, que possam testar a ferramenta e avaliar a hipótese;
- Apresentar e discutir novas técnicas de automatização de testes por meio de modelos de linguagem;
- Criação de um repositório com o código fonte do projeto.

1.4 Contribuições do Trabalho

O trabalho desenvolvido para esse projeto de diplomação traz consigo as seguintes contribuições:

- Uma nova ferramenta *open-source* capaz de auxiliar o projeto de sistemas de aquisição de sinais. Por meio de uma interface CLI, o usuário fornece o enunciado do projeto de interesse e, opcionalmente, responde a algumas perguntas. Seguindo uma metodologia

⁴ *Spectre Circuit Simulator* é um simulador elétrico, cuja proprietária e distribuidora é a *Cadence Design Systems*.

de síntese a nível de sistema com uma abordagem *top-down* dirigida por restrições, a ferramenta primeiro apresenta arquitetura da solução, para que, na sequência, forneça as implementações sugerida de cada bloco.

- Uma nova abordagem para automatizar a execução de aplicações que exijam interação humana. Visto que a ferramenta desenvolvida é, de forma simplificada, uma conversa entre um GPT e um usuário, é difícil automatizar a interação entre os agentes. Por meio de um GPT emulador, pode-se executar cenários de teste por meio de *prompts* de sistema específicos ao *testbench* desejado. Em adição a isso, faz-se uso de um segundo GPT para avaliar o resultado, visto que a saída também pode ser demasiadamente não-determinística para métodos convencionais.
- Quatro *testbenches* desenvolvidos para testar o desempenho de ferramentas cujo objetivo seja auxiliar o projeto de sistemas de aquisição de sinais. Os projetos exemplo escolhidos exigem em diferentes capacidades de resolução, algumas delas mais matemáticas, outras mais conceituais.

1.5 Considerações Éticas

Muito de se fala do impacto que a inteligência artificial trará à sociedade, tanto positivo, quanto negativo. Com a popularização do GPT, pessoas passaram a questionar a longevidade das suas profissões e de suas responsabilidades, de modo a criar uma espécie de alarmismo. Julgar essas preocupações como nada além de medo do desconhecido deve ser considerado descaso, visto que pesquisas da própria OpenAI (Eloundou *et al.*, 2023) indicam que existe sim, de fato, a perspectiva dos modelos inteligentes assumirem funções atualmente atribuídas a pessoas.

Não é a primeira vez que isso acontece na história. Todas as revoluções industriais anteriores⁵ trouxeram consigo a necessidade de ressignificar o conceito de profissional, de modo a se tornar uma fase quase que inevitável do seu ciclo de vida. Apesar das novas tecnologias permitirem automatizações na rotina de trabalho, o intelecto humano ainda possui seu valor imensurável, de forma que a sociedade deve, como um todo, compreender quais responsabilidades são dignas de serem ocupadas por pessoas. Entende-se que não é a evolução tecnológica que proporciona desigualdades laborais, mas sim o contexto no qual o todo está inserido.

Outra discussão a se levantar a respeito do presente trabalho é em relação ao impacto educacional. A ferramenta utiliza projetos da cadeira de Instrumentação A da UFRGS como *testbenches*, que pode levar ao questionamento desse projeto ser utilizado por alunos para que se obtenham vantagens injustas. Mesmo que a conclusão desse projeto

⁵ Considera-se que hoje vivemos a quarta revolução industrial (Schwab, 2017).

de diplomação fosse a de que a ferramenta está boa o suficiente para ser utilizada em projetos reais julga-se natural que novas tecnologias afetem o plano de ensino. É inevitável que estudantes aprendam métodos para facilitar seu trabalho, mas, dessa forma, é papel do sistema educacional se adaptar para que, em conjunto às ferramentas disponíveis, os alunos se tornem profissionais mais qualificados.

Em conclusão, compreende-se que é responsabilidade dos pesquisadores e profissionais abraçarem o desenvolvimento de novas tecnologias que podem contribuir positivamente para a área, mas de forma a sempre preservar o senso crítico, questionando a ética envolvida. O papel da academia é justamente o de fomentar a ciência por meio do pensamento crítico.

1.6 Estrutura do Trabalho

Os capítulos desse trabalho estão organizados da forma tradicional da qual muitas pesquisas fazem uso geralmente: inicia-se com uma revisão bibliográfica de conceitos relevantes ao projeto, apresenta-se a metodologia empregada e, por fim, os resultados são expostos e discutidos. No entanto, existem algumas observações a serem feitas para que se entenda as escolhas tomadas pelo autor.

A área de tecnologia, por si só, já é na sua maioria dominada por termos na língua inglesa. No entanto, GPTs trazem uma limitação adicional, pois foram originalmente treinadas apenas por meio de textos em inglês, de modo que é mais interessante ao trabalho que todos os *prompts* não sejam escritos em português. Assim, todos os *prompts* e mensagens que foram enviadas ao modelo conversacional foram escritas na língua inglesa e apresentadas dessa forma no trabalho, pois as expressões e terminologias originais são de suma importância. Além disso, algumas palavras novas, que ainda não possuem equivalência amplamente aceitas no português, como é o exemplo da própria palavra *prompt*, utilizada algumas vezes nessa seção, não serão traduzidas.

Em decorrência da quantidade de *prompts* que foram utilizadas para esse projeto, achou-se melhor incluí-los no apêndice, no intuito de não quebrar o fluxo de leitura da monografia. Além disso, o conjunto de informações e arquivos gerados ao fim de uma execução é grande demais para que esses estejam contidos no trabalho. Cada uma das 160 iterações possuem, no mínimo, 3 arquivos de texto, uma imagem e alguns arquivos de *logs*. Portanto, disponibilizou-se o endereço eletrônico do projeto para que essas informações possam ser acessadas, caso desejado.

2 Fundamentação Teórica

2.1 Modelos Transformadores

2.1.1 Contexto Histórico

Transformadores pré-treinados gerais, do inglês *Generative Pre-Trained Transformers* (GPT), dominam hoje a área de processamento de linguagem natural, do inglês *Natural Language Processing* (NLP). Contudo, o caminho percorrido até a ascensão dos GPTs é longo, e suas origens se encontram no fim da década de 40, antes mesmo do termo de inteligência artificial ser oficialmente utilizado pela primeira vez em um *Workshop* Dartmouth em 1955 (McCarthy *et al.*, 2006).

O motivo que levou os modelos de linguagem a ganharem tanta fama, é justamente suas habilidades de conversação, uma vez que conseguem responder de forma coerente quase qualquer questionamento. No entanto, o início das NLPs é marcado pelas pesquisas em tradução de máquina, *machine translation* (MT) (Jones, 1994), iniciadas no fim da década de 40 e publicadas no início dos anos 50 (Locke; Booth, 1955). O assunto se popularizou com diversos grupos de pesquisa surgindo nos Estados Unidos da América, na Europa e na antiga União Soviética, com a primeira conferência sobre MT sendo sediada no Instituto de Tecnologia de Massachusetts (MIT) em 1952 (Hutchins, 1997). Logo depois, em 1954, houve a exibição conhecida como *IBM-Georgetown Demonstration*, na qual mais de 60 frases foram traduzidas do russo para o inglês - um marco histórico na área de NLP. É importante ressaltar a falta de recursos existentes nessa época, quando comparadas com o que está disponível atualmente. Além dos computadores utilizarem o mecanismo de cartões perfurados, linguagens de alto nível não existiam, de forma que a programação era feita em *Assembly* (Jones, 1994).

Não demorou muito tempo até a área de NLP se entrelaçar com a de IA. O artigo *Computing machinery and intelligence* (Turing, 1950), predecessor de muitos dos estudos de MT, foi um dos primeiros a associar inteligência computacional com linguagem natural. Entre diversos assuntos, Turing discute a possibilidade de máquinas se comunicarem de maneira humana e propõe um teste (posteriormente denominado de teste de Turing), em que o computador tentaria ganhar em um jogo de imitação fingindo se comunicar como uma pessoa. No jogo, uma pessoa externa deve se comunicar textualmente com um humano e uma máquina e, por meio das mensagens trocadas, adivinhar qual é qual.

No entanto, em paralelo aos estudos de *machine translation*, na década de 60 iniciaram-se os esforços voltados aos modelos conversacionais. A ELIZA (Weizenbaum, 1966), é considerada um dos primeiros *chat-bots* criados e foi desenvolvida com o intuito

de agir como uma terapeuta. No início, os modelos inteligentes aplicados a NLP não utilizavam redes neurais, mas sim métodos baseado em regras linguísticas (Lo; Singh *et al.*, 2023). Muitas vezes essas não eram nem provenientes de estudos vinculados a área da computação, mas sim a pesquisas que abordavam conceitos de estrutura e sintaxe textual (Chomsky, 1957). O funcionamento de ELIZA era basicamente o de identificar palavras chaves e, com elas, providenciar respostas específicas, de modo a espelhar as mensagens do usuário. Essa abordagem exigia com que as regras fossem escritas manualmente e lidava pobremente com *outliers*.

Também em 1966, foi publicado o relatório da ALPAC ¹ referente à qualidade dos avanços na área de NLP, mais especificamente em MT (ALPAC, 1966). Esse relatório impactou fortemente a academia e causou drásticas reduções dos grupos de estudos nessa área em países de língua inglesa. Essa época é chamada por Poibeau (2017) de *grande pausa* e o autor relata que a mesma perdurou até 1990.

Apesar dos obstáculos, o NLP evoluiu muito ao longo das décadas, especialmente com o avanço de técnicas de aprendizado de máquina e com a popularização de modelos especialistas. Durante os anos 90, a IBM se tornou uma das pioneiras em implementar modelos estatísticos para as ferramentas de NLP, abandonando aos poucos as técnicas gramaticais *Chomskyanas* ² antes empregadas (Foote, 2023).

As LSTMs ³ foram introduzidas no fim do século XX por Hochreiter e Schmidhuber (1997), e logo se tornaram o estado da arte em NLP. O *chat-bot* Siri, apresentado em 2007, antes de ser adquirido pela Apple, utilizava LSTMs e redes neurais convolucionais (CNNs), e se tornou pioneiro na área de assistentes pessoais baseados em IA (Adamopoulou; Moussiades, 2020).

A grande virada de chave foi em 2017, com uma publicação realizada por um grupo da Google com a universidade de Toronto, chamada de *Attention is all you need* (Vaswani *et al.*, 2017). Esse trabalho apresentou ao mundo o conceito de transformadores, que permitiu que as LLMs que conhecemos hoje pudessem ser concebidas. Anterior a essa inédita arquitetura de *Deep Learning*, o estado da arte das NLPs utilizava RNNs, como LSTMs e GRUs (Chung *et al.*, 2014), e CNNs (Lecun *et al.*, 1998).

2.1.2 A Arquitetura e as Vantagens dos Transformadores

RNNs do tipo *Long Short Term Memory* (LSTM), que até 2017 eram consideradas o estado da arte em NLP, possuíam alguns problemas críticos que as impediram de competir

¹ *Automatic Language Processing Advisory Committee* foi um comitê de sete cientistas, cujo objetivo era avaliar o progresso na área de linguística computacional e, mais especificamente, de tradução de máquina.

² Termo dado às técnicas que se baseiam nas pesquisas linguísticas de Noam Chomsky.

³ Arquitetura de IA capaz de considerar aspectos temporais. A sigla, em tradução direta ao português, significa Memória de Curto Longo Prazo.

com os transformadores. Um dos fundamentos das redes neurais recorrentes, é a análise sequencial das entradas. Em termos simples, a arquitetura é uma série de redes *feed-forward* agrupadas sequencialmente, em que os modelos consomem as palavras uma por vez e vão armazenando informações na forma de estados escondidos (do inglês, *hidden states*). Em tese, esses estados guardam contexto e são o que permitem a geração de texto de saída. No entanto, essa abordagem é muito suscetível a explosão de gradiente ⁴ e sofre ao tentar armazenar dependências contextuais de textos grandes, *esquecendo* a relação com o que foi analisado em passos anteriores. Além disso, o fator sequencial prejudica drasticamente o desempenho, exigindo que diversos métodos de otimização sejam implementados para que essa arquitetura possa ser treinada em tempo hábil. Um desses métodos é o de truncar o algoritmo *backpropagation through time* (BPTT), de forma a somente incluir alguns nodos anteriores ao realizar a retropropagação (Jaeger, 2005).

CNNs, por outro lado, utilizam *kernels* para processar texto. Originalmente desenvolvidos para visão computacional, esses modelos até podem ser aplicados em NLP e são bons em detectar padrões dentro do espaço delimitado pelo *kernel* (Poria; Cambria; Gelbukh, 2016). Contudo, justamente devido ao tamanho fixo dessas janelas, a CNN tem dificuldade em entender o contexto geral, e falha ao relacionar as dependências textuais.

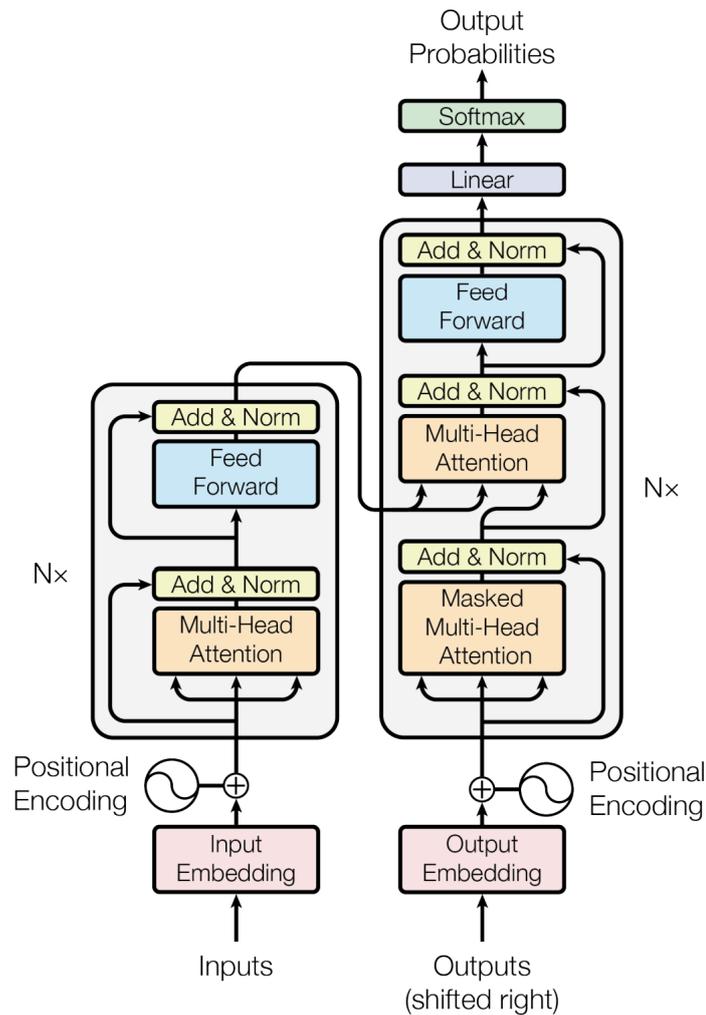
Os transformadores procuram solucionar o problema sequencial das RNNs e possuem uma abordagem de análise do texto inteiramente paralela. O mecanismo de auto-atenção, traduzido do inglês *self-attention*, busca entender como palavras ⁵ se relacionam umas com as outras e, com isso, atribui valores numéricos referentes a sua importância contextual. O algoritmo de atenção já era utilizado e conhecido (Bahdanau; Cho; Bengio, 2016) (Kim *et al.*, 2017), porém a contribuição do trabalho de Vaswani *et al.* (2017) está justamente na auto-atenção, cuja análise contextual é feita de forma reflexiva: cada palavra é analisada relativa às outras do mesmo texto.

A arquitetura original, proposta por Vaswani *et al.* (2017) pode ser vista na Figura 1 e foi principalmente desenvolvida para atuar em modelos de *machine translation*. O conjunto de blocos à esquerda da imagem são parte do conjunto codificador, que consome a entrada textual, e o conjunto à direita é o decodificador, que recebe como entrada a saída gerada até então.

⁴ Refere-se ao fenômeno em aprendizado de máquina em que os gradientes durante o treinamento tornam-se extremamente grandes, levando a instabilidade numérica e dificultando a convergência. Além disso, em arquiteturas com muitas camadas, a explosão de gradiente pode resultar na perda de informação das camadas iniciais. No contexto de NLP, seria como se as primeiras frases de um texto não fossem consideradas para estabelecer o contexto de um texto completo.

⁵ Por enquanto, será considerado que os modelos dividem o texto em palavras, mas isso é uma simplificação. Geralmente os modelos possuem seu próprio tipo de codificação, como *tokens*.

Figura 1 – Arquitetura proposta do Transformador.



Fonte: Vaswani *et al.* (2017)

O primeiro bloco a ser analisado é o de *embedding*, que converte a entrada textual em formato numérico, a fim de facilitar o processamento. Essa etapa, que se encontra logo no início do processo, consegue mapear palavras em um *embedding space* e, dessa forma, ao fornecer uma palavra, tem-se um vetor de saída. Esse espaço pode ser treinado para que o algoritmo possa corretamente capturar relações semânticas entre palavras.

No entanto, dependendo da ordem que se encontram na frase, palavras podem assumir significados diferentes. Visto que transformadores não processam dados de forma sequencial, uma codificação posicional incorpora ao mapeamento anterior a informação referente à ordem.

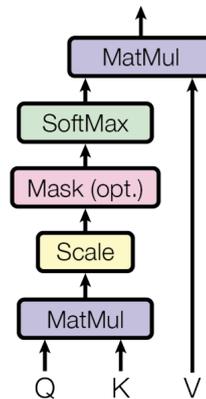
Adentrando mais no conceito de *self-attention*, esse mecanismo irá, para cada palavra, retornar um vetor que indica o quão forte é a relação dessa palavra com cada uma das outras. A dimensão é o número total de palavras, visto que cada posição é referente a uma das palavras fornecidas. Contudo, às vezes as palavras analisadas tendem a dar

mais valor a si mesmas, o que impede que informações relevantes sejam repassadas para os próximo bloco. Portanto, o *Multi-Head Attention* é uma série camadas paralelas de *self-attention* responsáveis por captar diferentes aspectos das relações entre cada par de palavras. Ao fim, a soma ponderada dos vetores é contabilizada, aumentando a densidade de informação contidas em cada vetor de atenção resultante.

O princípio matemático por trás do conceito de *self-attention* é um produto escalar que utiliza 3 vetores: *query* (Q), *key* (K) e *value* (V). A Figura 2 apresenta uma representação gráfica da equação a seguir:

$$Z = softmax\left(\frac{Q \cdot K^T}{\sqrt{\text{dimensão de } K, Q \text{ ou } V}}\right) \cdot V \quad (1)$$

Figura 2 – Diagrama de blocos representando o cálculo do vetor de atenção.



Fonte: Vaswani *et al.* (2017)

Em termos simples, Q e K encontram relações entre palavras, sendo que *query* em português é busca, e *key* é chave. O parâmetro de busca testará todas as chaves do texto de entrada para ponderar o quanto cada uma se encaixa no contexto. Esses dois vetores são otimizados durante treinamento, de forma que se procura entender quais chaves funcionam melhor para quais buscas. As pontuações de atenção são calculadas por meio de um produto escalar e são, posteriormente, normalizadas, dividindo-as pela raiz quadrada da dimensão dos vetores. Então, a função *softmax* é aplicada para converter as pontuações em probabilidades que somam 1.

O vetor V é referente a representações de palavras que serão ponderadas pela atenção (calculada entre Q e K) e resulta em uma representação de contexto enriquecido para cada palavra, para que se possa entender quais palavras subsequentes são mais prováveis de aparecer. Resumidamente e em outros termos, K e Q transformam o *embedding space* das palavras, de forma a melhor relacionar e distinguir contextos. Em sequência, o vetor

resultante é ponderado com o vetor V para que se possa inferir em quais regiões do espaço a próxima palavra está situada.

O bloco *Masked Multi-Head Attention* é, em grande parte, igual ao *Multi-Head* normal. A única diferença é que durante treinamento preditivo, ele ignora palavras do futuro ao gerar o vetor de atenção. Isso é feito atribuindo pesos insignificantes (como $-\infty$ antes da *softmax*) para garantir que apenas as informações das palavras anteriores ou atuais sejam utilizadas. O outro bloco de *self-attention*, que une a porção de entrada com a de saída, é o *encoder-decoder attention*. Neste, os vetores de K e V vêm da parte codificador (conjunto de blocos que se encontram na porção esquerda da figura 1) e o vetor Q vem do decodificador, que permite que cada posição no decodificador atenda a todas as posições do codificador, o que ajuda o modelo a focar em partes relevantes da entrada, ao gerar a saída.

Observam-se, também, dois blocos *feed-forward* (FNN), que são *Multi-Layer Perceptrons* (MLPs), seguidos de uma ativação ReLU, na qual se aplicam transformações não lineares aos dados, aumentando a capacidade do modelo de aprender representações complexas durante o treinamento. A camada linear, que se encontra no fim da arquitetura, é outra FNN, mas essa é encarregada de redimensionar o vetor para ser do tamanho de palavras possíveis de saída. Caso o modelo seja usado para traduzir palavras em inglês para português, esse bloco redimensionaria o vetor para ser do tamanho de palavras contidas no dicionário português. Em sequência, a função *Softmax* transforma esse vetor em uma distribuição probabilística.

2.1.3 Transformadores Pré-treinados Generativos

2.1.3.1 Arquitetura

Os Transformadores Pré-treinados Generativos são variações dos transformadores originais apresentados na subseção anterior. O funcionamento base é o mesmo, mas existem algumas diferenças importantes a serem destacadas.

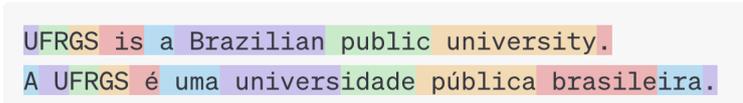
Por simplicidade, considerou-se até então que os transformadores mapeiam palavras ao *embedded space*⁶. Apesar de ser uma possibilidade, é mais comum que cada tipo de arquitetura empregue sua própria definição dessas unidades textuais, que são conhecidas como *tokens*. *Tokens* podem ser diretamente relacionadas a palavras - e, na maioria das vezes, são. Porém, mapear todas as palavras existentes e suas variações (maiúsculo, minúsculo, seguido de vírgula, etc), assim como as não existentes, em um espaço finito, é ineficiente. Dessa forma, o GPT realiza uma divisão por *tokens*, em que apenas as combinações de caracteres mais comuns, sejam eles especiais ou não, são mapeadas no

⁶ Uma representação matemática em que palavras ou entidades são codificadas como vetores em um espaço multidimensional de forma que suas posições refletem relações semânticas. Pode ser traduzido para espaço de incorporação.

espaço. Dessa forma, palavras grandes tendem a ser compostas por mais de um *token*, enquanto que palavras extremamente comuns, como artigos, possuem próprios *tokens*, independente da variação. A Figura 3 apresenta exemplos de como algumas frases são divididas.

Figura 3 – Divisão de texto em unidades de *tokens*.

Tokens	Characters
21	86



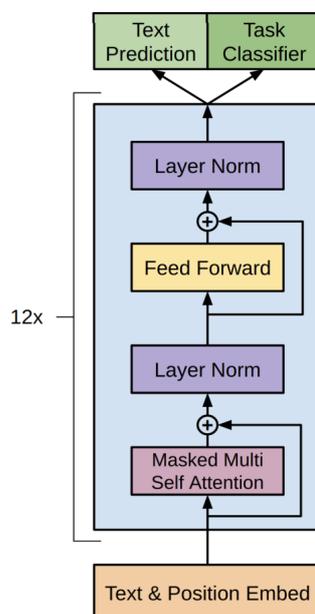
Fonte: retirado de OpenAI (2021)

Percebe-se que na primeira frase, "U" é um *token*, enquanto que na segunda, " U" é outro. O espaço já é suficiente para que possuam codificações diferentes, mesmo que apenas marginalmente. A partir desse ponto, será utilizado o termo *token*, em vez de *palavra*, para se referenciar à entidade granular processada pelo GPT.

Contudo, antes de adentrar na arquitetura empregada pela OpenAI - empresa responsável pelo desenvolvimento do GPT - é importante notar que somente detalhes de implementação até a versão GPT-3 são públicos. A OpenAI não divulgou arquiteturas, número de parâmetros ou qualquer outras especificações referentes aos modelos GPT-3.5-Turbo e GPT-4 (OPENAI, 2023a). Dessa forma, será apresentado o que sabe das versões iniciais do GPT, no entanto existe uma possibilidade das novas não seguirem esse exato formato.

A principal diferença entre o GPT e o transformador tradicional, é a ausência da parte codificadora. GPTs possuem o único intuito de geração de texto, enquanto que transformadores descritos por Vaswani *et al.* (2017) foram principalmente pensados para MT. Esses ainda conseguiriam gerar texto, mas a arquitetura não estava especializada para tal. A Figura 4 apresenta o diagrama de blocos da arquitetura do GPT, desenvolvida pela OpenAI.

Figura 4 – Arquitetura do GPT original.



Fonte: Radford *et al.* (2018)

Observa-se que, além de conter apenas o decodificador, o GPT faz uso apenas de um *Multi-Head Attention* por camada. Entretanto, fora algumas diferenças adicionais nas normalizações, as arquiteturas são conceitualmente parecidas e possuem as mesmas vantagens quando comparadas àquelas de outras famílias, como as RNNs.

Ao reduzir a quantidade de blocos e manter o fluxo mais linear, as camadas do GPT tornam-se facilmente cascadeáveis, de forma que a saída de uma camada, alimenta a entrada da próxima. Em questões de detalhes da arquitetura, os modelos GPT, GPT-2 e GPT-3 não possuem significativas variações, porém o número de camadas e, portanto, de parâmetros, foi significativamente crescendo (Radford *et al.*, 2019), (Brown *et al.*, 2020). O GPT original contava apenas com 12 camadas, enquanto o GPT-2 possui 48 e a maior variação do GPT-3, 96.

Em vez de usar um codificador para tratar a entrada e um decodificador para processar o texto gerado de saída, o GPT considera que o texto de entrada é, de fato, a saída gerada até o momento. Assim, dispensa-se o uso do codificador. Por exemplo, ao se providenciar a entrada *para colar grau é necessário apresentar um projeto de*, essa frase seria alimentada para o decodificador como se tivesse sido gerada pela saída até o momento, e o *token* escolhido seria, provavelmente, *diplomação*. Por meio de *tokens* de posição ou *tokens* especiais, o modelo até consegue diferenciar a entrada, mas conforme a saída é gerada, ambos são incorporados uns aos outros, conforme mostra o exemplo abaixo.

- A UFRGS é \rightarrow uma
- A UFRGS é uma \rightarrow universidade
- A UFRGS é uma universidade \rightarrow pública

2.1.3.2 Técnicas de Treinamento

Esses modelos podem ser pré-treinados, visto que basta fornecer uma grande massa de texto e instruir o preenchimento de lacunas. Esse tipo de treinamento é comumente conhecido como auto-supervisionado, uma vez que o próprio modelo aprende a prever o próximo *token* com base nos anteriores, sem rótulos externos. Contudo, tendo em mente que as *labels* não são anotadas e validadas por humanos, às vezes o treinamento é classificado como não-supervisionado (Radford *et al.*, 2018). Esse pré-treinamento possui o objetivo de ensinar aos modelos os conceitos linguísticos fundamentais, além de facilitar a implementação futura de outras técnicas de treinamento, como *fine-tuning* ou variações da *meta-learning*.

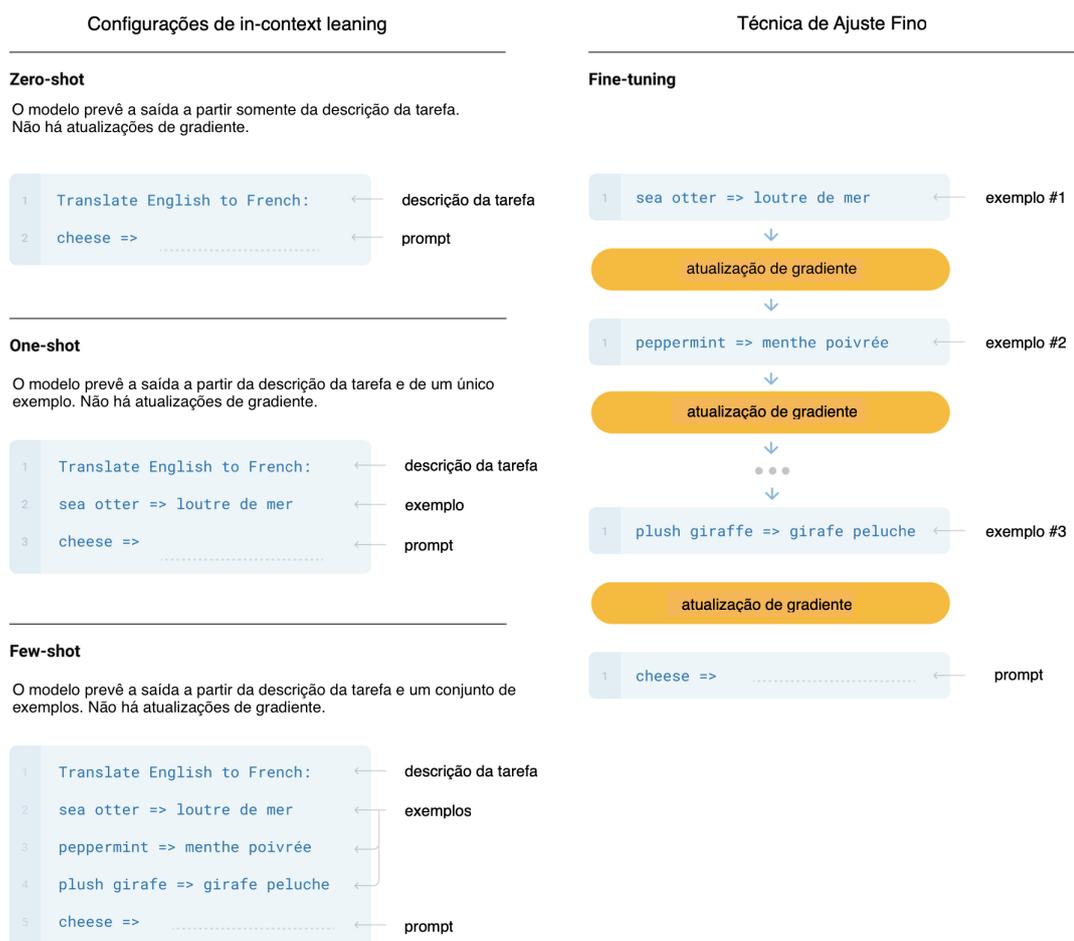
Todos os GPTs, independente da sua versão, passam primeiro pelo pré-treinamento não-supervisionado. No entanto, para que o modelo consiga desempenhar melhor, outras técnicas devem ser utilizadas na sequência. O GPT original utilizava o ajuste fino, um método supervisionado que procura treinar o modelo para tarefas específicas, como sumarização de texto, tradução, etc. Assim, o GPT aprende como lidar com situações específicas, melhorando os resultados para essas tarefas. Contudo, existem problemas com essa abordagem, principalmente relacionados ao tamanho do *dataset* que é necessário - para o GPT era preciso uma média de 100 mil amostras por tarefa. Vale lembrar que esse treinamento é supervisionado, implicando na composição - ou no mínimo validação - manual desses dados. Outro problema com o *fine-tuning* é a suscetibilidade a *over-fitting* caso o *dataset* seja grande ou específico demais. Além disso, tarefas que não estejam no escopo do ajuste fino desempenham pior, prejudicando a capacidade do modelo de corretamente responder frases de múltiplos contextos, como, por exemplo, realizar cálculos matemáticos no meio de uma pergunta sobre alocação de recursos financeiros.

Visto que o objetivo da OpenAI é desenvolver um modelo generalista que não é especializado em tarefas específicas, o GPT-2 abandonou o ajuste fino em troca do *zero-shot learning* (Radford *et al.*, 2019). Em vez de providenciar um *dataset* referente a tarefas que o modelo deva aprender, o *zero-shot* é um tipo de aprendizado *in-context*, ou seja, ele receberá um *prompt* de entrada e uma descrição da tarefa. O contexto adicional permite que o modelo entenda qual parte do seu pré-treinamento deve ser ativada. Esse método não atualiza o gradiente, portanto, ele não serve de fato como um treinamento adicional. No entanto, o nome inclui o termo *learning*, dado que o modelo, em tese, aprende na hora de inferência que tipo de resposta ele deve providenciar dado o contexto.

Considerando que essa abordagem é bastante dependente das capacidades aprendidas durante o pré-treinamento para que se possam obter resultados melhores, a escala da arquitetura deve aumentar. É por esse motivo, principalmente, que o número de camadas passou de 12 para 48, da versão 1 para a 2. Contudo, a troca de *fine-tuning* por *zero-shot* não trouxe melhorias ao desempenho, pelo contrário, trouxe pioras (Radford *et al.*, 2019). Apesar de ser incapaz de manter o mesmo nível para todos os contextos, o ajuste fino realmente melhorava o modelo para algumas tarefas.

Com o intuito de continuamente aumentar a performance dos modelos, o GPT-3 implementou novas variações à técnica de aprendizado *in-context*. Em vez de apenas fornecer o *prompt* e a descrição da tarefa, incluiu-se, também, uma série de exemplos. Caso apenas um único exemplo seja passado, tem-se o *one-shot*, e caso sejam múltiplos, tem-se o *few-shot* (Brown *et al.*, 2020). Assim como o *zero-shot*, esses métodos não atualizam o gradiente. A figura 5 apresenta a diferença entre o ajuste fino e os métodos *in-context*.

Figura 5 – Comparação entre os métodos *in-context* com o *fine-tuning*.



Fonte: adaptado de Brown *et al.* (2020).

Assim como a transição entre GPT para GPT-2 exigiu um grande aumento da escala do modelo, também foi necessário aumentar o número de camadas na versão GPT-3. A maior variação da terceira geração contou com 96 camadas de decodificadores, o que implica em 175 bilhões de parâmetros (Brown *et al.*, 2020).

O *dataset* de treinamento do GPT-3 foi constituído por 4 fontes de dados (Brown *et al.*, 2020), a principal é a Common Crawl (2007), que oferece um *web scrape*⁷ da internet, do período desejado. No entanto, uma vez que esses dados podem não ter a qualidade esperada, a OpenAI aplicou alguns filtros que reduziram o tamanho do *dataset* para cerca de 410 bilhões de *tokens*. Além do *Common Crawl*, utilizaram-se dados da *Wikipédia* (Wikipedia contributors, 2004) e de mais duas fontes de dados: *WebText2* (Gao *et al.*, 2020), que contém dados do *Reddit*⁸ e; *Books1* e *Books2*, que contém texto extraído de livros do site *Smashwords* (Coker, 2009). Entretanto, a soma do número de *tokens* dessas bases não somam a 100 bilhões, de forma que o *Common Crawl* constitui a maior parte dos dados de treinamento.

Com esse conhecimento, pode-se agora diferenciar os modelos conhecidos como GPT - discutidos até então - com o ChatGPT. Originalmente, o ChatGPT usou uma versão do GPT-3 como motor, porém com uma diferença adicional: o ajuste fino. Como o objetivo original do modelo era servir como um *chat-bot* capaz de manter conversas com usuários humanos, a etapa de treinamento supervisionada foi incluída para melhorar a tarefa conversacional.

2.1.3.3 Modelos Disponíveis

A OpenAI oferece uma vasta gama de modelos, cada uma com diferentes objetivos e características. Antes das versões mais recentes dos GPTs, as funcionalidades de geração de texto, imagens e sons estavam claramente delimitadas. Contudo, os modelos mais novos possuem integração com a família DALL-E (Betker *et al.*, 2023) e TTS (OPENAI, 2023e).

Como discutido anteriormente, não se tem conhecimento aprofundado sobre a arquitetura dos GPTs posteriores ao GPT-3. No entanto, a melhoria das funcionalidades oferecidas são evidentes e permitem que o usuário escolha adequadamente o melhor modelo para seu projeto. A Tabela 1 compara as características de cada modelo a partir do GPT-3.5-turbo, pois os anteriores já estão considerados depreciados.

⁷ esse termo se refere a quando é retirado o conteúdo da internet para montar um banco de dados, pode ser traduzido diretamente para *raspagem da internet*.

⁸ Rede social estilo fórum (REDDIT INC., 2005).

Tabela 1 – Comparação das funcionalidades dos modelos GPT, a partir do GPT-3.5-Turbo.

Modelo	Janela de contexto	Máximo de saída	tokens	Dataset de treinamento
gpt-3.5-turbo	4096	-		Até Setembro 2021
gpt-3.5-turbo-16k	16385	-		Até Setembro 2021
gpt-4	8192	-		Até Setembro 2021
gpt-4-32k	32768	-		Até Setembro 2021
gpt-4-1106-preview	128000	4096		Até Abril 2023

Fonte: extraído de OpenAI (2023b).

O modelo denominado *gpt-4-1106-preview*, o integrante mais recente da família GPT, é uma versão beta do GPT-4-Turbo, e foi lançado no dia 6 de novembro de 2023. Seu *dataset* de pré-treinamento foi retirado de dados da internet mais recentes, até abril de 2023, único dos GPTs cujos dados não são limitados a 2021. Além disso, seu contexto (maior número de *tokens* de entrada e de saída somados permitidos em uma interação) é aproximadamente 4 vezes maior que o GPT-4-32k, que até então possuía a maior janela de contexto. A única desvantagem que o GPT-4-Turbo possui é o limite de *tokens* de saída, que está fixo em 4096. Acredita-se, no entanto, que isso é devido ao fato de ser uma versão ainda não produtizada.

A Tabela 2 apresenta o preço, em dólares, para utilizar os modelos. O valor é contabilizado em lotes de 1000 *tokens* cada, com custos mais elevados aos de saída. Sabe-se que informações monetárias são de pouco valor, dado que são altamente voláteis. No entanto, foram importantes para o presente trabalho, uma vez que influenciaram na escolha do modelo.

Tabela 2 – Precificação dos modelos GPT-3.5 e GPT-4.

Modelo	Custo por 1000 tokens de entrada	Custo por 1000 tokens de saída
gpt-3.5-turbo	\$0.0015	\$0.002
gpt-3.5-turbo-16k	\$0.003	\$0.004
gpt-4	\$0.03	\$0.06
gpt-4-32k	\$0.06	\$0.12
gpt-4-1106-preview	\$0.01	\$0.03

Fonte: extraído de OpenAI (2023d).

Em termos monetários, a família GPT-4 é a de custo mais elevado e, de acordo com a OpenAI (2023c), procuraram reduzi-lo com o modelo GPT-4-Turbo.

2.1.3.4 API

Os modelos GPT são acessíveis apenas via interfaces de programação de aplicativos (API, do inglês, *application programming interface*), portanto, a principal ferramenta para a execução do projeto foi a API de conclusão de texto conversacional.

Devido à grande oferta de modelos - cada uma com suas diferentes capacidades - a OpenAI possui uma vasta oferta de *endpoints*⁹. Contudo, visto que o objetivo do trabalho é em relação aos modelos conversacionais, mais especificamente os modelos modernos não-deprecados, apenas um único *endpoint* será abordado nessa revisão: a API de conclusão de texto conversacional (em inglês, *Chat-Completion*). Abaixo estão listadas algumas características dessa API.

- Endereço: <https://api.openai.com/v1/chat/completions>
- Método: POST
- Autorização do tipo *Bearer Token*

Ao obter a inscrição às APIs da OpenAI, é fornecido ao usuário um *Bearer Token*¹⁰ de acesso, que deve ser enviado como um *Header* na requisição. Para utilizar esse *endpoint*, deve-se também incluir no corpo da requisição um objeto contendo, no mínimo, o modelo desejado e as mensagens. Em 2.1 é possível ver um exemplo.

Listing 2.1 – Exemplo de corpo de requisição para a API de conclusão de texto da OpenAI

```
1 {
2   "model": "gpt-3.5-turbo",
3   "temperature": 0.1,
4   "messages": [
5     {
6       "role": "system",
7       "content": "You are a helpful assistant."
8     },
9     {
10      "role": "user",
11      "content": "Hello! What are you?"
12    },
13    {
14      "role": "assistant",
```

⁹ Refere-se aos pontos de entrada ou saída de uma comunicação entre diferentes sistemas, como em APIs.

¹⁰ Um tipo de *token* de acesso utilizado em sistemas de autenticação, onde o próprio *token* é suficiente para provar a identidade do usuário, sem a necessidade de apresentar outras credenciais, como usuário e senha.

```
15     "content": "Hello! I am GPT, a large language model developed by
16         OpenAI."
17     },
18     {
19         "role": "user",
20         "content": "Great! I need help with a math problem."
21     }
22 ]
```

Percebe-se que é possível enviar tanto a mensagem do usuário, quanto as mensagens de sistema e o histórico da conversa. O *prompt* de sistema é necessário para definir as instruções base do modelo, que influenciam o método e o estilo de respostas. O histórico de mensagens é necessário devido ao fato do modelo não possuir memória e, para que o modelo leve em consideração as trocas de mensagens prévias, é necessário reenviá-las.

Além dos campos evidenciados no exemplo, é possível, ainda, enviar um campo chamado *tools* (anteriormente denominamos *functions*, que está sendo deprecado), permitindo com que o usuário envie as definições de ferramentas externas. No momento, apenas funções estão disponíveis, mas desejam expandir para outros tipos de ferramentas integradas. Tendo em mente que essa funcionalidade não foi utilizada no projeto, ela não será aprofundada.

O campo de temperatura - *temperature*, no inglês - não é mandatório, porém influencia demasiadamente a consistência das respostas. A temperatura indica o quão determinístico será o texto gerado. Ou seja, quanto maior for o valor, maior será a diferença entre respostas a mesma pergunta. Esse campo é do tipo *float*, e é limitado entre 0 e 2 inclusivo.

O campo *top_p* também pode ser passado como parâmetro, a fim de manipular o comportamento determinístico do modelo. Porém, ele atua em cima da probabilidade de cada *token* a ser escolhido. Como discutido na subseção anterior, as LLMs elencam os *tokens* com maior probabilidade de seguirem o atual. Dessa forma, o campo *top_p* indica um valor percentual que limitará o modelo a apenas escolher, por exemplo, um *token* entre os 10% mais prováveis, caso 0,1 for atribuído ao campo.

A API oferece uma grande quantidade de campos para personalizar e moldar a resposta do modelo, no entanto a maior parte não foi utilizada para o presente projeto. Por conseguinte, segue abaixo uma exposição de todos os possíveis campos da requisição do *endpoint* de *chat-completion*.

- *model* - string identificadora do modelo de LLM a ser utilizada.

- *messages* - uma lista de objetos de mensagens, que podem ser compostas por mensagens de tipo *system*, *user*, *assistant* ou *tool*.
- *temperature* - valor entre 0 e 2 para manipular o comportamento determinístico do modelo, sendo 0 o mais determinístico.
- *top_p* - um valor entre 0 e 1 que indica qual porcentagem dos *tokens* mais prováveis considerar na escolha.
- *n* - número inteiro de iterações de respostas a serem geradas.
- *stream* - define se o a resposta será enviado pelo método *stream*, que permite que o texto aparenta estar sendo gerado aos poucos.
- *stop* - uma *string*, ou um vetor de *strings*, que resultará na finalização da mensagem, caso o modelo escolher gerá-las.
- *max_tokens* - define o número máximo de *tokens* a ser gerado. Cada modelo já possui um valor máximo, porém esse campo permite reduzi-lo ainda mais.
- *frequency_penalty* - numero entre -2,0 e 2,0 que penaliza ou incentiva a repetição de *tokens*. Valores positivos penalizam.
- *logit_bias* - um mapa que permite que se associe um *bias* entre -100 e 100 a *tokens* específicos, resultando em uma maior probabilidade de aparência caso o valor for positivo.
- *user* - uma identificação do usuário, que permite que a OpenAI controle a origem de mensagens potencialmente maliciosas.
- *logprobs* - um parâmetro booleano para que ser retorne adicionalmente a probabilidade associada a cada *token* de saída gerado.
- *top_logprobs* - um inteiro entre 0 e 5 que retorna os *x tokens* mais prováveis para cada posição.
- *seed* - uma funcionalidade em versão beta para que se possa reproduzir exatamente a mesma saída, caso o *seed* for o mesmo. Determinismo não é garantido.
- *response_format* - apenas disponível para o gpt-4-1106-preview, permite que se especifique o formato desejado de saída. Apenas disponível *text* ou *json*.
- *tools* - uma lista de tipos ferramentas para disponibilizar ao modelo.
- *tool_choice* - utilizado para fornecer as definições das ferramentas que podem ser utilizadas pelo modelo.

- *function_call* - deprecado e substituído pelo parâmetro *tool_choice*.
- *functions* - deprecado e substituído pelo parâmetro *tools*.

O objeto de resposta, por sua vez, é mais conciso. Além dos metadados, como identificação, *timestamp* de criação e nome do objeto, a resposta é composta por dois principais campos: a resposta do modelo em si e a quantidade de *tokens* usados. Em 2.2 pode-se ver um exemplo do corpo de resposta da requisição.

Listing 2.2 – Resposta JSON da API de conclusão de texto da OpenAI.

```
1 {
2   "id": "chatcmpl-123",
3   "object": "chat.completion",
4   "created": 1677652288,
5   "system_fingerprint": "fp_44709d6fcb",
6   "choices": [{
7     "index": 0,
8     "message": {
9       "role": "assistant",
10      "content": "\n\nHello there, how may I assist you today?",
11    },,
12    "logprobs": null,
13    "finish_reason": "stop"
14  },
15  "usage": {
16    "prompt_tokens": 9,
17    "completion_tokens": 12,
18    "total_tokens": 21
19  }
20 }
```

Nota-se que um objeto *usage* é recebido, o que indica a quantidade de *tokens* utilizados pelo usuário, a quantidade de *tokens* respondidas pelo modelo e a soma dos dois. Essa métrica é de importante para poder avaliar devidamente o custo de uso da aplicação. No entanto, a principal informação recebida está na lista de objetos denominada *choices*, que representa a resposta composta pelo modelo e o motivo da geração de texto ter sido interrompida. Esse campo é uma lista justamente pela possibilidade de especificar, por meio do parâmetro *n*, o número de respostas a serem fornecidas pelo modelo. O campo *finish_reason* indica se o modelo interrompeu a geração de texto por motivos de: extrapolação do número de *tokens* máximo; por querer chamar uma *tool*; pelo conteúdo

ter sido omitido pelo filtro da OpenAI; ou por simplesmente ter sido finalizado. Por fim, o *logprobs* só não estará vazio caso o parâmetro *logprobs* ou *top_logprobs* seja fornecido na requisição.

2.1.4 Engenharia de *Prompt*

Modelos baseados em transformadores são altamente contextuais. Assim, a formulação do *prompt* a ser enviado por modelo é de suma importância, podendo ser o fator determinante para se obter uma resposta adequada. Essa composição de texto, colocada em alguns casos como supostamente simples é, na verdade, tão vital ao uso de LLMs, que tem se tornado uma área nova de estudo por si só, chamada de engenharia de *prompt*.

Quando analisam-se, especificamente, os modelos GPT da OpenAI, considera-se dois principais tipos de *prompts*: usuário e sistema. A mensagem de sistema é única e deve ser sempre a primeira no vetor enviado na requisição. Essa é responsável por providenciar uma série de instruções base às quais o modelo deve tentar atender, similar ao treinamento *in-context* visto na seção 2.1.3.2. A maioria das estratégias de engenharia de *prompt* que serão abordadas visam a maximizar a eficácia dessa mensagem.

O outro tipo de *prompt* que pode ser manipulado é o de usuário. Contudo, nem sempre isso é possível, pois, em muitos casos, essa mensagem é de fato proveniente de um usuário real, que pode não possuir conhecimento das premissas da engenharia de *prompt*. Pode-se, alternativamente, concatenar a mensagem do usuário com um *prompt* pré-definido, mas deve-se avaliar a real necessidade e vantagem disso, já que um texto estruturado será misturado com algo desconhecido.

Engenharia de *prompt* é um campo ainda em desenvolvimento. Algumas das técnicas são mais intuitivas e fáceis de implementar, em contrapartida, outras são menos convencionais e requerem boas justificativas. O presente trabalho não necessita de métodos complexos, e portanto utiliza-se apenas de estratégias mais intuitivas, aceitas como sendo eficazes no que se propõem.

2.1.4.1 Personas

Provavelmente a estratégia mais comum e facilmente implementável de engenharia de *prompt* é a *persona* (White *et al.*, 2023a), (White *et al.*, 2023b). Se fosse necessário fazer uma pergunta matemática a alguém, entende-se que a melhor opção seria escolher uma pessoa formada na área das exatas. Por outro lado, se a dúvida fosse médica, um profissional da área da saúde seria a melhor alternativa. A técnica de *persona* não é nada além disso. Pede-se que um modelo assuma alguma personalidade, de modo que suas respostas sejam respondidas por meio do viés especificado. Diferentemente de algumas outras estruturas de *prompts*, a técnica de *persona* é aplicável em virtualmente qualquer

caso, inclusive em combinação com outras estratégias. Para montar um *prompt* desses, basta definir qual papel o modelo deve assumir.

Listing 2.3 – Exemplo da estratégia de persona para construção dos prompts.

Você é um engenheiro eletricista especialista na área de instrumentação.

Você nunca recomenda componentes sem saber as características do projeto.

No exemplo acima é compartilhado com o modelo que ele deve assumir a persona de um engenheiro eletricista especialista em instrumentação. A informação providenciada já é suficiente para que, se o usuário pedir um recomendação de filtro, o modelo entenda que o filtro em questão é relacionado a eletrônica e não a café, por exemplo. A segunda frase no *prompt* indica que, em vez de recomendar um filtro genérico, o modelo procuraria entender mais do problema em questão antes de dar uma resposta.

Em resumo, personas fornecem o contexto inicial e a *personalidade* do modelo. Esse padrão de *prompt* é tão comum, que todos os projetos discutidos na seção 2.5 utilizam uma forma ou outra dessa estratégia.

2.1.4.2 Perguntas

Escopo é um fator determinante na qualidade das respostas das LLMs. Ao montar os *prompts*, algo que se deve ter muito cuidado é evitar alucinações, termo dado para quando os modelos providenciam informações erradas, mesmo que pareçam estar convencidas da sua veracidade. Existem várias formas de aumentar o risco de alucinações, dentre as mais comuns estão o escopo excessivamente grande e informações conflitantes (Giray, 2023).

A não ser que a pergunta a ser feita ao modelo esteja muito mal formulada, é difícil conseguir fazer com que as LLMs se neguem a responder uma pergunta. Portanto, embora às vezes não existam informações suficientes para que uma resposta adequada seja fornecida, o modelo tentará, mesmo que o conteúdo seja pura alucinação. Isso também ocorre frequentemente quando o usuário insere dados conflitantes, para os quais uma resposta correta nem poderia existir.

Problemas como esse são especialmente comuns em projetos de engenharia de software, em que GPTs são utilizados para desenvolver código. Para que a solução seja viável e útil, muitas vezes a informação providenciada no início não é suficiente. Para tal, a estratégia de *prompt* de perguntas é interessante. Antes de pedir para que o modelo de fato monte a solução do problema, a LLM pode assumir o papel de formulador de perguntas, para que lacunas ou inconsistências possam ser esclarecidas. Um detalhe importante dessa implementação é garantir que o número máximo de perguntas esteja explicitado, visto que sem um teto, o modelo tende a perguntar o máximo, mesmo que o escopo do

questionamento fuja um pouco do objetivo. O ideal é que apenas perguntas relevantes e necessárias sejam feitas. O exemplo a seguir apresenta um método de implementar o padrão de perguntas.

Listing 2.4 – Exemplo da estratégia de perguntas para construção dos prompts.

Você é um engenheiro eletricista.

Você receberá um enunciado de um projeto de eletrônica, porém, antes de providenciar uma solução, você deve esclarecer pontos que possam ter ficado em aberto sobre a implementação.

Pergunte ao usuário até 5 perguntas, cujas respostas lhe permitirão desenvolver corretamente a solução.

Nota-se que, além do *prompt* de perguntas, incluiu-se a técnica de personas.

2.1.4.3 *Templates*

Os padrões de *prompts* apresentados até então podem ser utilizados para qualquer aplicação das LLMs. Contudo, poucos padrões são tão pertinentes para a implementação em *software* quanto o de *templates*. Respostas fornecidas pelo modelo não seguem nenhum padrão rígido, de forma que é impossível prever com garantia suficiente qual será o formato do texto.

Caso utilize-se a estratégia de perguntas, mencionado anteriormente, sem delimitar exatamente a formatação desejada, o GPT pode preferir ora enumerá-las, ora embuti-las em texto. Assim, a manipulação via código se torna desnecessariamente impraticável.

Entretanto, caso um *template* seja fornecido, de modo que apenas as lacunas devam ser preenchidas pelo modelo, é muito mais fácil implementar o GPT em código confiavelmente. Existem diversas formas de realizar isso, principalmente com JSON e *Markdown*. Nessa abordagem, é vital que fique absolutamente claro ao modelo a necessidade de seguir rigorosamente a formatação.

Listing 2.5 – Exemplo da estratégia de *templates* para construção dos prompts.

Você é um engenheiro eletricista.

Você receberá um enunciado de um projeto de eletrônica, porém, antes de providenciar uma solução, você deve esclarecer pontos que possam ter ficado em aberto sobre a implementação.

Pergunte ao usuário até 5 perguntas, cujas respostas lhe permitirão desenvolver corretamente a solução.

Você deve seguir o template abaixo de forma rigorosa, apenas substituindo os placeholders em maiúsculo pelas perguntas.

· · ·

1. PERGUNTA_1

2. PERGUNTA_2

3. PERGUNTA_3

4. PERGUNTA_4

5. PERGUNTA_5

· · ·

O exemplo providenciado une as três estratégias de *prompts* vistas nessa subseção. O template escolhido no exemplo é o de um bloco *Markdown*.

2.2 Sistemas de Aquisição de Sinais

Vive-se hoje a era *Big Data*, na qual nossa simples pegada digital possui valor na economia. Antes, considerava-se que o mundo vivia sua era digital, mas, de certa forma isso se mantém verdade; a única diferença é que a tecnologia digital tem sido tão normalizada em todos os aspectos do cotidiano, que pode passar despercebido.

A área de sistemas de aquisição de sinais faz interface por todas as *eras* pelas quais já foram, e estão sendo, vividas. Sinais são medidos analogicamente e convertidos para o domínio digital, na forma de dados. Entende-se que o *data* no termo *Big Data* geralmente é associada mais à dados de usuários, como tendências consumidoras. Contudo, com avanços de *IoT*, por exemplo, todos os sinais, até os mais simples, tornam-se dados.

2.2.1 Fluxo Clássico de Projeto

O fluxo de projeto depende muito do objetivo e, portanto, do mensurando. De modo geral, conforme Taylor (1997), os elementos de um sistema de aquisição de dados se enquadram nas seguintes categorias:

- **Sensores** ou transdutores, que compõem o primeiro estágio do sistema, responsáveis por converterem o mensurando (pressão, temperatura, entre outros) em sinais elétricos.
- **Condicionamento**, que englobam todos os estágios encarregados de manipular o sinal, de forma a prepará-lo para ser amostrado pelo ADC.
- O **Filtro *Anti-Aliasing***, apesar de poder ser considerado parte do condicionamento do sinal, cumpre uma função imprescindível em qualquer sistema de aquisição de dados. Sem essa etapa, o sinal passa a estar sujeito a *aliasing*¹¹, impossibilitando a reconstrução do sinal no domínio digital.
- **Sample and Hold** é um bloco que, em conjunto a um multiplexador, permite a amostragem de diversos canais simultaneamente. Além disso, pode ser utilizado antes do ADC para segurar o valor a ser convertido até que o processo acabe.
- O **Multiplexador** atua como um seletor de canal, permitindo que sistemas mais complexos, compostos por múltiplos sensores iguais ou diferentes, possam ser adquiridos pelo mesmo ADC.
- O **Conversor Analógico-Digital** é o bloco ao redor do qual todos outros componentes giram, pois a partir dessa etapa, o sinal passa a estar no domínio digital. Dada sua importância, é necessário conhecer o mensurando, o condicionamento e o processo de aquisição subsequente para que se possa escolher apropriadamente o ADC. Na maioria dos sistemas, o maior custo energético é atribuído ao conversor.

O autor segue a lista com mais etapas de sistemas de aquisição, como *clock*, *Random Access Memory* (RAM), conversor digital-analógico (DAC), interfaces e *bus*. Entretanto, o último elemento de interesse ao presente trabalho é o ADC, visto que o foco está na aquisição do sinal analógico e subsequente conversão ao domínio digital, não necessariamente ao processamento digital.

No livro *Sensors and Signal Conditioning* (Pallas-Areny; Webster, 2012), os autores associam à etapa de condicionamento de sinal a amplificação, filtragem, *impedance matching*, *level shifting*, modulação e demodulação. Entende-se *impedance matching* como a manipulação de igualar a impedância de saída de um bloco com a de entrada do subsequente, de modo a garantir compatibilidade. O *level shifter* também cumpre essa função, porém garantindo que o nível de tensão está devidamente ajustado para que possa servir de entrada à etapa seguinte.

¹¹ Fenômeno presente em processamento de sinais digitais em que diferentes sinais são indistinguíveis uns dos outros devido à taxa de amostragem inadequada, resultando em informações falsas ou distorcidas.

2.2.2 Síntese a Nível de Sistema

Otimização está presente em qualquer área de estudo, pois sempre existe alguma característica que desejamos minimizar (ou maximizar), como custo e tempo. Com o aumento da escala dos sistemas eletrônicos, foi necessário desenvolver metodologias de síntese capazes de melhor estruturar o processo, trazendo consigo uma maior eficácia.

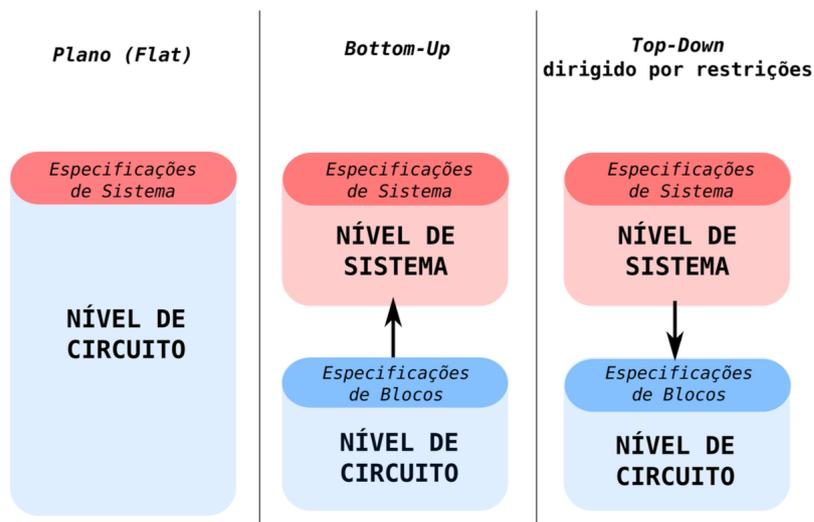
Síntese nada mais é do que a transformações de restrições em implementações (Gerstlauer *et al.*, 2009). Assim, procura-se entender quais os requisitos do sistema, para que as escolhas certas a nível de topologias, circuitos e arquitetura possam ser tomadas.

A síntese a nível de circuito trata em diretamente dimensionar os circuitos dadas as restrições. Dessa forma, no contexto de circuitos analógicos, escolhem-se os componentes, ou seja, transistores, capacitores, resistores, indutores e diodos, baseados no que se sabe dos requisitos de projeto (Weber, 2015). Essa metodologia possui diferentes abordagens, como a baseada em conhecimento e a baseada em otimização, porém, de modo geral, entende-se que existem *trade-offs* ao escolher a síntese a nível de circuito. Tendo em mente que as restrições são diretamente consideradas durante o projeto, o trabalho necessário pode não escalar bem, possivelmente inviabilizando essa metodologia para grandes projetos. Por outro lado, dado que as escolhas são diretamente a nível de componente, uma das vantagens é que há mais controle sobre a implementação.

A síntese a nível de sistema trata o projeto de outra forma, de modo a otimizar e generalizar o processo. Em vez de tomar as restrições e diretamente sintetizá-las, essa metodologia exige que uma dada sequência de passos sejam tomadas. Essas etapas são definidas pela abordagem escolhida, dentre as quais está a *top-down* dirigida por restrições, utilizada nesse trabalho.

A Figura 6 apresenta de forma visual os passos presentes em algumas das diferentes abordagens da síntese a nível de sistema.

Figura 6 – Diferentes abordagens da síntese a nível de sistema.



Fonte: adaptado de Weber (2015)

A estratégia plana leva todas as especificações do sistema e sintetiza o projeto, sem direcionamento claro. Gielen, McConaghy e Eeckelaert (2005) descreve essa abordagem como tentar projetar um ADC a nível de transistor, ou seja, acaba por se tornar complexo demais.

A abordagem *Bottom-up* procura entender individualmente as especificações de cada bloco para, na sequência, considerar as restrições de sistema. Apesar da maior organização, quando comparada ao plano, a estratégia *Bottom-up* traz consigo alguns problemas. Devido à ordem dos processos, pode-se descobrir apenas após a síntese de alguns blocos que nem todas as especificações de sistema estão satisfeitas. Não há método claro de resolver isso, levando a uma possível necessidade de recomeço completo (Gielen; McConaghy; Eeckelaert, 2005).

Por fim, tem-se a abordagem *top-down* dirigida por restrições. Esta procura inverter o processo de síntese descrito na *Bottom-up*, ou seja, primeiro são analisadas as especificações a nível de sistema, para que se possa projetar a arquitetura. Dessa forma, estabelecem-se os requisitos de cada bloco envolvido, sem que haja a possibilidade de descobrir posteriormente uma incompatibilidade, na teoria.

A partir das especificações individuais de cada componente da arquitetura, é possível repetir esse processo, em que o limite de granulação é critério da pessoa projetista. Dessa maneira, isolam-se os requisitos de cada bloco, reduzindo o escopo. Por exemplo, caso as especificações mínimas de um bloco amplificador tenham sido decididas em nível de sistema, é possível fazer o projeto do amplificador em nível de circuito sem considerar simultaneamente todas as variáveis dos demais blocos. No entanto, circuitos analógicos

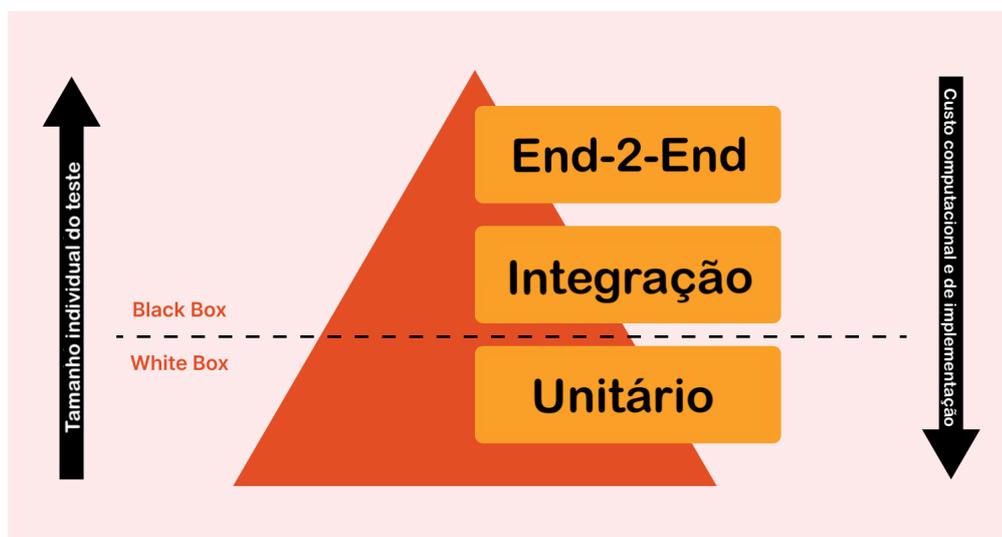
não permitem uma abstração completa, visto que as impedâncias dos demais blocos que interfaceiam com o amplificador serão importantes. Logo, isso deve ser considerado também durante a especificação dos blocos.

2.3 Testes Unitários e a Pirâmide de Testes

Testes são imprescindíveis para desenvolver um código confiável. No entanto, a garantia de funcionamento do que foi desenvolvido é somente um reflexo dos cenários de teste implementados. A estratégia de cobertura deve considerar uma série de fatores, mas, de forma geral, a pirâmide de testes, introduzida na sua primeira versão por Cohn (2010), é um bom ponto de partida.

A Figura 7 apresenta uma adaptação mais usual da pirâmide de testes. Pensada com o intuito de auxiliar times ágeis de desenvolvimento (Sutherland; Sutherland, 2014), a representação triangular permite que se compreenda o *trade-off* de cada tipo de teste, de modo que se possa traçar um plano eficiente de cobertura de uma aplicação qualquer.

Figura 7 – Pirâmide de teste.



Fonte: adaptado de Bhardwaj (2023).

Testes unitários cumprem a função de testar o menor componente da aplicação, que, dependendo da paradigma de projeto, tende a ser uma classe ou uma função. Dessa forma, entende-se que a base da pirâmide é formada por uma massa de pequenos testes, cada um com o objetivo de validar um mínimo escopo.

Idealmente, a maior parte da cobertura deve ser feita por testes unitários, deixando apenas a conectividade, a integração dos componentes e as questões de infraestrutura a encargo dos demais tipos de testes. Estes são mais custosos de desenvolver e de executar, pois geralmente requerem um ambiente dedicado.

Testes manuais não estão representados pela pirâmide, pois, idealmente, não devem ser utilizados como uma ferramenta de validação da funcionalidade da aplicação. Simulações ou testes exploratórios podem assumir processos manuais, porém, de forma geral, a base de cobertura do funcionamento do código em si deve ser atribuído aos testes descritos na pirâmide.

Sobre a implementação dos testes unitários, considerando que a menor unidade de código é a função, basta definir os valores esperados de saída dadas as entradas. Nem todas as funções podem - ou devem - ser testadas. Um componente encarregado apenas por chamar outros componentes, sem que exista funcionalidade própria, não deve ser sujeito a teste unitário, visto que o que estará de fato sendo coberto é a pilha de funções abaixo dessa.

2.4 Mann–Whitney U-Test

Existe uma gama de métodos estatísticos que devem ser considerados quando se deseja analisar e comparar amostras. No entanto, conforme as características dos dados são evidenciadas, é necessário saber escolher qual método utilizar. O *Mann-Whitney U-Test* (Mann; Whitney, 1947) possui uma metodologia que o torna uma boa opção para dados não-paramétricos, ou seja, amostras que não seguem distribuição alguma.

Conhecido também como *Wilcoxon Rank-Sum Test*, o U-test não faz suposição alguma da distribuição dos dados, pois o primeiro passo é ordenar esses, de forma que a classificação abstraia a informação contida no dado em si. A hipótese nula H_0 é a de que não existe diferença significativa entre a distribuição das amostras de dois grupos distintos. Dessa forma, esse método permite apenas a análise comparativa entre duas populações.

Uma vez que o primeiro passo é ordenar as amostras, pressupõe-se que os dados possuem características ordinais, caso contrário, não seria possível aplicar esse método. Em situações de um ou mais empates, deve-se encontrar o ponto intermediário das classificações e atribuir esse valor a todas as amostras contidas no empate.

Em seguida, deve-se encontrar o valor de U para cada uma das populações por meio da equação

$$U_A = n_A \cdot n_B + \frac{n_A \cdot (n_A + 1)}{2} - T_A, \quad (2)$$

em que se supõe que A e B são os índices referentes a cada um dos grupos, que n é o número de amostras e que T_A é a soma das classificações de A. Esse cálculo deve ser feito, também, para a população B, mas com os índices trocados. Em sequência, é utilizado apenas o valor mínimo entre os valores de U , conforme a expressão a seguir:

$$U = \min(U_A, U_B) . \quad (3)$$

O valor de U obtido é suficiente para testar H_0 utilizando a tabela referência (ver apêndice B) e, se o valor calculado for menor que o tabelado, rejeita-se H_0 . No entanto, em casos cujos dados possuem muitas repetições ou cujas populações possuem mais que 20 amostras, é recomendado encontrar o valor z para testar a hipótese. Para tal, deve-se calcular a média 4 e o desvio padrão 5, supondo que a hipótese H_0 é verdadeira.

$$\mu_U = \frac{n_A \cdot n_B}{2} \quad (4)$$

$$\sigma_U = \sqrt{\frac{n_A \cdot n_B \cdot (n_A + n_B + 1)}{12}} \quad (5)$$

Caso hajam muitos valores repetidos no grupo amostral, existem duas abordagens para lidar com o problema. A primeira é encontrar algum outro método de desempatar os valores; a outra é utilizar a versão expandida da equação do desvio padrão, descrita por

$$\sigma_U = \sqrt{\frac{n_A \cdot n_B \cdot (n_A + n_B + 1)}{12} - \frac{n_A \cdot n_B \sum_{k=1}^K (t_k^3 + t_k)}{12 \cdot n \cdot (n - 1)}} , \quad (6)$$

em que t_k é o número de empates na classificação k -ésima e K é o conjunto de classificações em que houveram empates (Lehmann, 1975).

Assim, pode-se calcular o escore padronizado z com

$$z = \frac{U - \mu_U}{\sigma_U} . \quad (7)$$

Se o número de amostras por grupo for maior que 20, a significância pode ser aproximada por uma distribuição normal utilizando o p -value. Caso contrário, pode-se utilizar o próprio valor referência de z -value para comprovar a hipótese, que, para um $\alpha = 0,05$ é 1,96. Assim, caso o valor z calculado seja maior que 1,96, conclui-se que os dois grupos amostrais apresentam diferença significativa e a hipótese H_0 é rejeitada.

Percebe-se que para testar a hipótese com os valores de U , tem-se

$$\text{se } U_{calc} < U_{tab} \quad \text{rejeita-se } H_0 . \quad (8)$$

Enquanto que, para testar por meio do z -value, tem-se a comparação oposta, dada por

$$\text{se } z_{calc} > z_{tab} \quad \text{rejeita-se } H_0 . \quad (9)$$

2.5 Estado da Arte

2.5.1 Projetos Baseadas em LLMs

GPTs encontraram uma grande adesão em diversas áreas da engenharia, dentre estas está a geração de código. O projeto *open-source* mais popular¹² para essa função é o *gpt-engineer* de Osika (2023), que oferece funcionalidades para geração e melhoria de código. A ferramenta exige um enunciado inicial e, na sequência, pede ao modelo que este faça perguntas ao usuário para entender os requisitos do projeto.

Osika (2023) oferece um exemplo em que a ferramenta desenvolve um jogo *snake*, para o qual são gerados arquivos Python, JavaScript, CSS e HTML.

Também com o objetivo de geração de código, o *gpt-pilot* (PYTHAGORA-IO, 2023) permite que GPTs sejam utilizados para o desenvolvimento de aplicações de forma supervisionada. O fluxo da ferramenta procura emular os passos tomados por times de engenharia. Assim, primeiro são esclarecidos os requisitos de negócio; em seguida, os técnicos e; por fim, o projeto é dividido em tarefas que serão utilizadas para de fato gerar o código.

No entanto, GPTs podem ser utilizados para facilitar projetos de outras formas. Em projetos grandes, independente se estes são da área de engenharia ou não, é necessário que os modelos generativos consigam levar em consideração uma série de especificações, que podem estar inseridas no meio de textos. Em muitos casos, LLMs podem se perder no contexto e não conseguir absorver todas as informações que lhe foram passadas. O projeto LLMLingua (Jiang *et al.*, 2023) (LongLLMLingua... , 2023) é uma ferramenta para condensar *prompts*, reduzindo o tamanho desses, otimizando custo e resultados. Nem sempre a saída desse aplicativo é um texto humanamente coeso, isso acontece porque aproveita-se do que LLMs consideram *tokens* relevantes.

Também com o objetivo de otimizar *tokens*, é interessante que se possua uma ferramenta de síntese de texto. O projeto *TopicGPT* (Pham *et al.*, 2023) fornece tais capacidades, de forma a fornecer o principais tópicos relacionados, com funcionalidades adicionais de criar hierarquias granulares do assunto do texto de entrada. A pesquisa apresenta um método próprio de avaliação que utiliza os tópicos presentes em artigos da Wikipédia (Wikipedia contributors, 2004), como base comparativa.

No entanto, outro método de avaliar respostas às soluções *open-ended*, ou seja, perguntas que não possuam resposta única, é por meio do projeto *JudgeLM* (Zhu; Wang; Wang, 2023). Uma das partes mais desafiadoras de realizar pesquisa com LLMs é conseguir encontrar métodos confiáveis de avaliação, portanto, o projeto utiliza uma etapa de treinamento supervisionado de ajuste fino.

¹² Com base no número de *stars* (estrelas) que o repositório possui na plataforma do GitHub.

2.5.2 Ferramentas pra Síntese de Projetos

A ferramenta desenvolvida pela Texas Instruments *Filter Pro* (Bishop, 2002) permite que usuários forneçam as especificações do filtro desejado e recebam maiores detalhes de modelos, implementações e parâmetros do modelo. Diferentemente dos outros trabalhos citados, o Filter Pro não inclui componente inteligente, mas auxilia o projeto de forma clássica e confiável.

2.5.3 Trabalhos que Implementam LLM na Área de Engenharia Elétrica

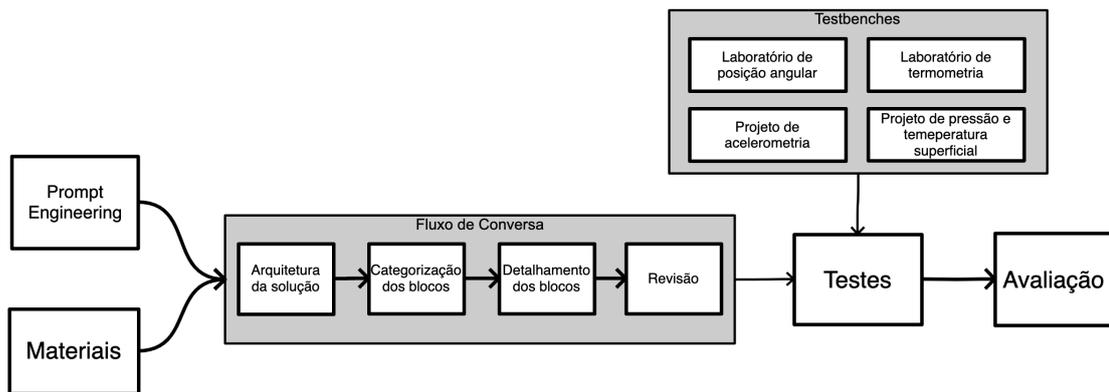
A literatura ainda não possui uma grande quantidade de trabalhos aplicados diretamente à área de engenharia elétrica. No entanto, Blocklove *et al.* (2023) discute o processo de síntese de *chips*, que inicia com a análise textual dos requisitos de projetos, para que pessoas projetistas possam traduzir essas especificações em implementações. O Chip-Chat se diz ser a primeira vez em que um modelo conversacional co-autora a síntese de um chip, visto que os autores da pesquisa fabricam e testam o projeto desenvolvido. ChatGPT-4 foi o modelo utilizado.

Com o intuito de conseguir auxílio ao projeto de sistemas embarcados, Tafferner *et al.* (2023) procuraram descobrir os potenciais do ChatGPT para providenciar especificações de controladores, modelos de sensores, fluxo de síntese (*hardware* e *software*) e referências bibliográficas para o projeto de uma automatização residencial. Conclui-se que, apesar de algumas informações coerentes a respeito da escolha de controladores e de algumas outras especificações, o ChatGPT responde com inconsistências que precisam ser validadas por profissionais, não tornando-o apto a assumir essa função ainda. No entanto, os autores entendem que, no futuro, novos modelos serão utilizados como uma ferramenta adicional na síntese de projetos.

3 Metodologia

A metodologia do trabalho possui uma série etapas, uma vez que não apenas a ferramenta é uma contribuição do trabalho, mas os métodos de teste automatizados também. O fluxo de funcionamento da metodologia pode ser observado na Figura 8.

Figura 8 – Fluxo da metodologia experimental.



Fonte: o Autor, 2024.

Individualmente, as etapas de compõem a metodologia da ferramenta são:

1. **Materiais:** esta seção apresenta os materiais utilizados ao longo do trabalho, como linguagens de desenvolvimento, bibliotecas, modelo de LLM, entre outros apresentados na seção 3.1.
2. **Engenharia de *Prompt*:** aqui são discutidas as técnicas utilizadas para a composição dos *prompts* do projeto, apresentados em mais detalhes na seção 3.2.
3. **Fluxo de conversa:** essa seção aborda o método na qual a ferramenta utilizará o GPT, de modo a apresentar como as técnicas de engenharia de *prompt* são empregadas para guiar o modelo a compor uma implementação ao problema. Este fluxo é explicado aprofundadamente na seção 3.3.
4. **Testes:** essa etapa apresenta os testes utilizados para o projeto, como testes unitários e automatizados, conforme a seção 3.4.
5. ***Testbenches*:** essa etapa apresenta os conjunto de *testbenches* selecionados e adaptados para serem testados na ferramenta, descritos em detalhe na seção 3.5.
6. **Avaliação:** aqui são discutidos os métodos escolhidos para avaliar o desempenho da ferramenta e dos próprios testes automatizados, apresentados na seção 3.6.1.

3.1 Materiais e Ferramentas

3.1.1 Hardware Utilizado

Para o desenvolvimento e execução do projeto utilizou-se um computador MacBook Pro 16, cujas especificações técnicas são:

- CPU de 2,3GHz 8-core Intel Core i9
- Memória RAM de 16GB 2666MHz DDR4
- GPU MD Radeon Pro 5500M com 4GB de GDDR6

No entanto, para que se possa treinar e manter uma LLM da escala do GPT-4, é necessário um poder computacional muito maior. Por questões de sigilo, as especificações técnicas do *hardware* utilizado pela OpenAI não foram divulgadas. Contudo, conforme o artigo da própria OpenAI (Brown *et al.*, 2020), o modelo GPT-3 contava com cerca de 175 bilhões de parâmetros, o que permite que se tenha uma ideia da magnitude computacional necessária.

3.1.2 Software Utilizado

3.1.2.1 Modelos de Linguagem

Para esse trabalho, limitou-se o escopo apenas para os modelos de texto desenvolvidos pela OpenAI (ver seção 2.1.3.3), que são protagonistas na popularização das LLMs. Fez-se essa escolha por ser a ferramenta mais amplamente utilizada no mercado no presente momento.

Dentre todos os modelos oferecidos, selecionou-se o *gpt-4-1106-preview*, conhecido como GPT-4-Turbo, que é o modelo mais recente da OpenAI, lançado no dia 6 de Novembro de 2023. Percebe-se pelo nome do modelo que ele ainda está em fase beta, mas tendo em mente que os resultados parciais foram melhores com esse modelo e que, até o presente momento, o seu preço é significativamente menor (ver Tabela 2), decidiu-se por utilizar essa versão.

Destaca-se que o GPT-4-Turbo possui dados mais atualizados, além de ter um tamanho de contexto superior aos seus antecessores. No entanto, o único problema é que esse modelo é o único a possuir um limite no número de *tokens* de saída. Apesar de ser possivelmente limitante em algumas aplicações, para o presente projeto isso não é um problema, já que é no número de *tokens* de entrada que os valores podem ser grandes.

A partir de agora, por simplicidade, será utilizada apenas a sigla GPT para se referenciar ao modelo da OpenAI utilizado no trabalho. No entanto, entende-se que a versão subentendida é o GPT-4-Turbo.

3.1.2.2 Linguagem de Desenvolvimento

Visto que o modelo é exposto por meio de uma API, pode-se escolher, na teoria, qualquer linguagem de programação capaz de suportar comunicação HTTP. No entanto, existem bibliotecas que abstraem a camada de comunicação e permitem que o desenvolvimento seja composto apenas por chamadas de funções. A OpenAI oferece uma biblioteca *open-source* na linguagem *Python 3* (Rossum; Drake, 2009) com plena integração a todas as funcionalidades que a empresa oferece por meio de suas APIs (ver seção 2.1.3.4). No entanto, por motivos de preferência, escolheu-se desenvolver o projeto na linguagem de programação *Go* (Griesemer; Pike; Thompson, 2009). Devido a sua já existente e crescente popularidade, existem bibliotecas em *Go* que também abstraem a API oferecida pela OpenAI, com a diferença de que essas não são mantidas pela própria empresa. Apesar de existirem vantagens de se utilizar bibliotecas mantidas pela própria fornecedora da API, o projeto em *Go* é bastante popular e conta com constante contribuição da comunidade.

Além da biblioteca que realiza a interface à API da OpenAI, utilizaram-se outros módulos para desde auxiliar com *logs*, até destacar palavras em diferentes cores para uma melhor experiência do usuário. Utilizou-se, também, linguagens de desenvolvimento auxiliares, como o *GNU Make* (Feldman, 1976) para controle de executáveis, e o *DOT* (Ellson *et al.*, 2004) para descrição de gráficos (abstraído em *Go* pela biblioteca *go-graphviz*). Para uma visão geral dessas ferramentas e suas respectivas versões, veja a tabela 3.

Tabela 3 – Ferramentas computacionais utilizadas.

Ferramenta Computacional	Versão
Go	1.20.5
go-openai	1.17.9
go-graphviz	0.1.2
go-color	1.4.1
zap	1.26.0
godotenv	1.5.1
Testify	1.8.4
GNU Make	3.81

Fonte: o Autor, 2024.

A biblioteca *testify* foi utilizada para implementar os testes unitários e permite que, para uma única função, uma série de cenários possam ser construídos. Dentro destes, são fornecidos os valores de entrada e os de saída esperados, para garantir que são iguais.

Destaca-se, também, o projeto *chat-with-gpt*, que é uma UI para a API da OpenAI. Apesar dessa ferramenta não estar embutida na aplicação desse projeto, ela foi utilizada no início do desenvolvimento para testes, pois a interface oferece um método rápido e fácil para realizar provas de conceito. Entretanto, ela não é mais mantida pelo autor e, dessa forma, não suporta novas as versões e configurações do GPT.

3.2 Engenharia de *Prompt*

Os *prompts* têm se tornado a principal ferramenta ao se utilizar LLMs, motivo pela qual surgiu a nova área denominada *Engenharia de Prompt*, que procura entender como os modelos se comportam perante diferentes formatos de texto.

A engenharia de *prompt* foi essencial para o presente trabalho, visto que é imprescindível que se aproveite ao máximo a capacidade das LLMs, em vez de tirar conclusões precipitadas decorrentes da má estruturação dos *prompts*. Contudo, não foi necessário utilizar uma ampla gama de padrões de texto, visto que com apenas alguns já foi possível atingir os objetivos.

Embutido em todos os *prompts* de sistema fornecidos ao GPT ao longo do processo, inclusive naqueles utilizados para perguntas, estão os padrões de persona (ver seção 2.1.4.1). Dessa forma, é fornecido um norte inicial a respeito de com qual viés o modelo deve responder. Para essa ferramenta, a *persona* escolhida foi a de um *engenheiro de instrumentação eletrônica analógica especialista*.

Além disso, o uso de *templates*, discutido na seção 2.1.4.3, também presentes em todos os *prompts* de sistema, exigem que a resposta esteja inserida em um bloco *Markdown*, que simplifica a implementação em código.

No total, sem contar aqueles destinados a testes, a ferramenta conta com 24 *prompts*, 11 das quais são de sistema e os remanescentes, de usuário. Todos podem ser vistos no apêndice A.

3.3 Fluxo de Conversa

3.3.1 Visão Geral

Devido a natureza generalista dos modelos de linguagem, não existe restrição alguma referente à formatação e conteúdo das perguntas a serem feitas. No entanto, quando se deseja utilizar o modelo para resolver problemas específicos, como é o caso desse presente trabalho, a escolha dos *prompts* é essencial para determinar a qualidade, a consistência e a formatação das respostas. Além do uso dos *prompts*, a ordem na qual as perguntas devem ser feitas e da manipulação do histórico de conversa são ferramentas poderosas no desenvolvimento de projetos com LLMs.

A alucinação ¹ é um problema inerente do uso de LLMs, a ponto de ser impossível removê-la. No entanto, medidas podem ser tomadas para reduzir tanto a probabilidade de alucinações, quanto a gravidade destas. Uma das principais causas desse efeito é o escopo excessivamente amplo da requisição, de forma que o modelo fica encarregado de diversas

¹ Termo dado a quando modelos de linguagem fornecem informação errônea como se fosse verdade.

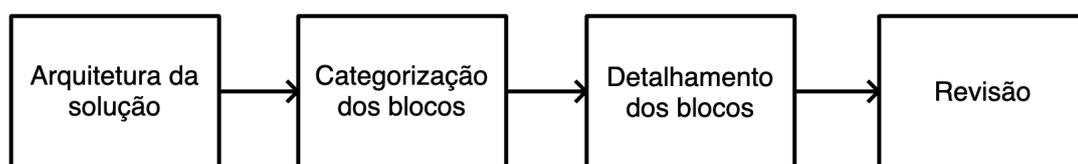
tarefas de uma única vez. Como as LLMs não possuem memória, isto é, apenas inferem *tokens* subsequentes a partir de um contexto, simplesmente pedir para que o GPT projete toda a arquitetura e detalhamento de um sistema de aquisição de sinais é impraticável. A separação em etapas é necessária.

Outro fator que reforça a inviabilidade da abordagem simples e direta é a possível falta de informações. A habilidade do modelo reconhecer lacunas no enunciado fornecido é imprescindível, pois nem sempre contém todos os detalhes necessários para implementação. Por exemplo, na maioria das vezes, o usuário não possui acesso a todos os sensores disponíveis no mercado, portanto, preferências pessoais também devem ser repassadas ao modelo inteligente.

Devido aos pontos discutidos, caso a ferramenta desenvolvida nesse trabalho apenas enviasse o enunciado ao GPT, sem nenhuma estratégia de *prompt* ou de fluxo de conversa, o projeto retornaria, no melhor caso possível, uma reescrita da entrada da entrada (sem aprofundamento ou adição de informação útil ao projeto), e no pior caso, uma solução completamente errônea. Por outro lado, entende-se que o projeto perde sentido se a ferramenta exige uma quantidade excessiva de informações do usuário.

Dessa forma, o fluxo da conversa que ocorre dentro da ferramenta possui diversas etapas e detalhes. De modo geral, o trabalho pode ser resumido em quatro principais blocos: desenvolvimento da arquitetura da solução, categorização interna dos blocos, detalhamento de cada bloco e, por fim, uma revisão da solução completa. A Figura 9 contém o diagrama em blocos da visão geral das etapas.

Figura 9 – Diagrama de blocos da visão geral do fluxo de conversa proposto.



Fonte: o Autor, 2024.

Tanto a etapa de desenvolvimento da arquitetura, quanto a de detalhamento dos blocos, são compostas por uma série de interações do usuário com o modelo GPT, com algumas poucas trocas entre o modelo e a ferramenta em si. Em contrapartida, as etapas de categorização e revisão não possuem interação com o usuário, pois cumprem funções internas da ferramenta.

As próximas subseções abordarão com um maior aprofundamento o processo descrito acima, assim como as justificativas de cada escolha. Todos os *prompts* utilizados no durante o fluxo da ferramenta, sejam de sistema, sejam de usuário, podem ser vistos no apêndice A do trabalho.

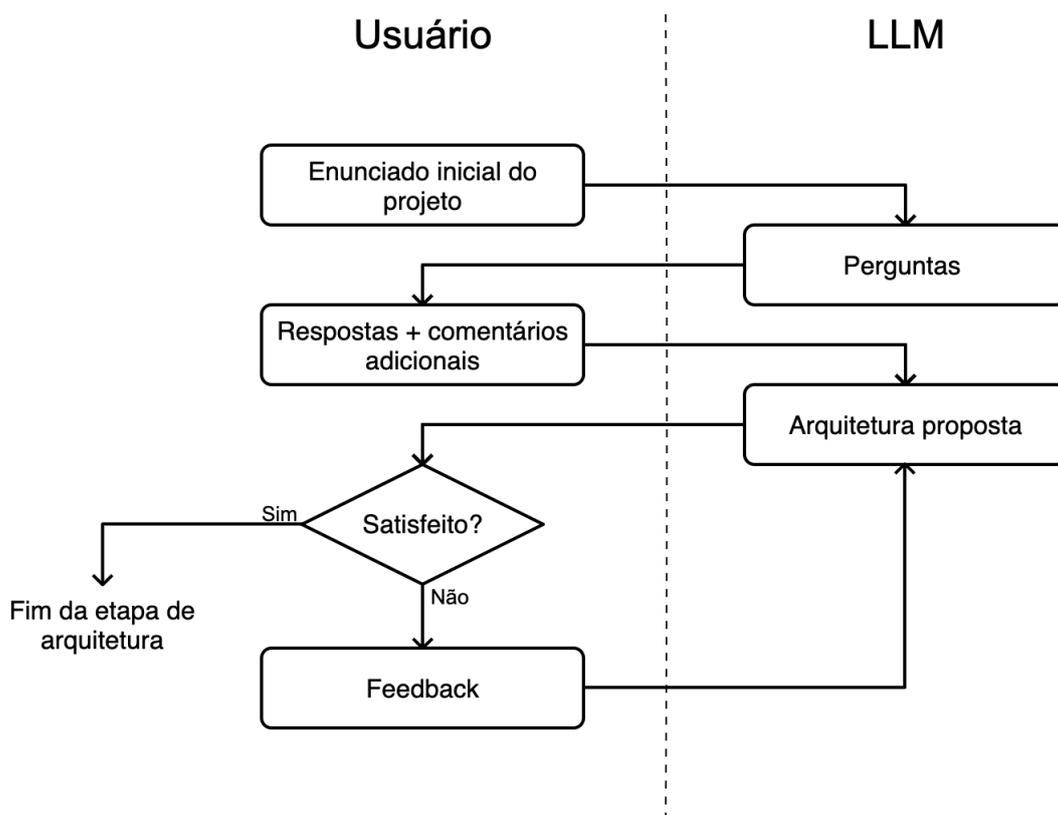
3.3.2 Desenvolvimento da Arquitetura

A abordagem *top-down* dirigida por restrições estabelece, em um momento inicial, a solução em nível de sistema, que fornece a arquitetura geral do projeto. Portanto, a primeira etapa da ferramenta possui o objetivo de definir o diagrama de blocos da solução, além de providenciar textos explicativos referentes a cada escolha.

Tendo em mente que podem não estar definidos todos requisitos de projeto, essa etapa não procura se aprofundar nas topologias e nos detalhes de cada bloco. Isso não reduz a importância da etapa de arquitetura. Pelo contrário, a arquitetura dita o fluxo subsequente de execução, de modo que erros no diagrama de blocos serão repassados aos detalhamentos, dificultando a obtenção de resultados bons.

As interações presentes nessa etapa do projeto - que são apenas entre o modelo e o usuário - estão apresentadas na Figura 10.

Figura 10 – Fluxo de conversa da etapa de arquitetura.



Fonte: o Autor, 2024.

Apesar das enormes capacidades das LLMs, os modelos não conseguem fornecer soluções complexas a partir de informações limitadas. Existe, portanto, uma necessidade natural de especificar alguns detalhes do projeto. Por outro lado, uma ferramenta que exige

um grande quantidade de informações rapidamente perde utilidade, pois se torna apenas uma ferramenta que traduz o enunciado em um diagrama de blocos e fornece textos a partir de informações já providenciadas.

Justamente para mitigar esses riscos, foi implementado um passo de perguntas, na qual o modelo receberá o enunciado inicial fornecido pelo usuário e formulará até 5 perguntas. O objetivo dessas perguntas é identificar a ausência de informações essenciais para a diagramação, estabelecer o tipo de sensor desejado para o projeto e entender se o usuário possui preferências de domínio analógico ou digital. O motivo pelo qual existe a necessidade de perguntar sobre o sensor e o domínio é que estes pontos são vitais e determinarão todo percurso da solução.

O usuário, em sequência, terá de responder as perguntas, mesmo que seja para dizer que não sabe responder ou não tem preferência de implementação. Além disso, haverá um espaço para incluir comentários adicionais que o usuário deseje informar ao modelo.

Com essas respostas, além de um *prompt* de sistema específico, o modelo será instruído a formular o diagrama de blocos. Antes de exibir os resultados, o diagrama de blocos, que é fornecido por meio da linguagem `.DOT`, é compilado pela ferramenta em formato SVG e PNG. Caso a compilação falhe por alguma questão de sintaxe, o programa automaticamente envia uma mensagem informando o erro ao modelo e tenta compilar a revisão fornecida. O aplicativo admite até 3 revisões antes de reportar o erro ao usuário e encerrar o processo.

Com o diagrama corretamente descrito e com as explicações fornecidas, a ferramenta pergunta ao usuário se a resposta está satisfatória. Caso não esteja, é solicitado ao usuário um comentário sobre o que deve ser alterado, que é concatenada a uma mensagem padrão e enviada ao GPT. Assim que o usuário receber a nova solução, a pergunta de satisfação é novamente feita, de modo que esse *loop* segue indefinidamente até que a resposta seja julgada satisfatória. Durante a etapa de *feedback*, todo histórico de conversa é mantido e enviado ao modelo, porém, assim que a arquitetura é aprovada pelo usuário, todas as interações referentes às correções são removidas. Isso impede com que o número de *tokens* acabe por superar o limite máximo permitido pelo modelo, conforme apresentado na subseção 2.1.3.3. Além disso, evitam-se que informações incorretas - mesmo que corrigidas - sejam enviadas ao GPT em todas as futuras requisições.

3.3.3 Categorização dos Blocos

Essa etapa consiste na categorização dos blocos da arquitetura, para otimizar o detalhamento por vir. Os modelos de linguagem, especialmente quando utilizados para questões técnicas, precisam de um maior direcionamento para evitar com que as respostas

fiquem genéricas demais e sem conteúdo, assim como é feita no *in-context learning* (ver seção 2.1.3.2).

A solução encontrada para maximizar as chances do modelo fornecer uma solução viável é a de categorizar os diferentes blocos da arquitetura para que, na etapa de detalhamento, os *prompts* de sistema sejam específicos. Evidentemente, não existe uma forma de garantir com que todos os blocos se enquadrem em alguma das categorias, porém, visto que o escopo está fechado para sistemas de aquisição de sinais, é possível estabelecer algumas categorias que tendem a aparecer mais nessa área, as quais estão expostas na lista abaixo:

- Sensor
- Condicionamento de sinal
- Amplificação
- Filtragem
- Outros tipos de condicionamento
- Medição direta
- Conversão analógico-digital
- Processamento digital
- Outros

O motivo pela qual foi incluído tanto uma categoria para *condicionamento de sinal*, quanto para *amplificação* e *filtragem*, é para que os *prompts* possam ser construídos com instruções específicas. É possível explicitamente solicitar a frequência de corte para categorias do tipo filtro, mas nem sempre isso seria aplicável para amplificadores. As categorias *Condicionamento de sinal* e *Outros tipos de condicionamento* atuam como categorias genéricas para blocos de condicionamento que não se encaixaram nos outros. A diferença entre essas categorias é sutil, mas as instruções presentes no *Condicionamento de sinal* são mais diretas ao solicitar sugestões de topologias e seus valores. Apesar da possível interseção de escopo, a existência dessas duas categorias não prejudica o trabalho. A categoria de *Outros*, cumpre o papel de último recurso, caso o bloco não possa ser classificado com nenhuma outra categoria. É obrigatório que todas as etapas da arquitetura sejam categorizadas.

Algumas das categorias possuem um escopo mais fechado, como amplificação, filtragem e conversor analógico digital (ADC). Por outro lado, processamento digital, abrange uma gama de áreas. No entanto, lembra-se que o foco está no domínio analógico

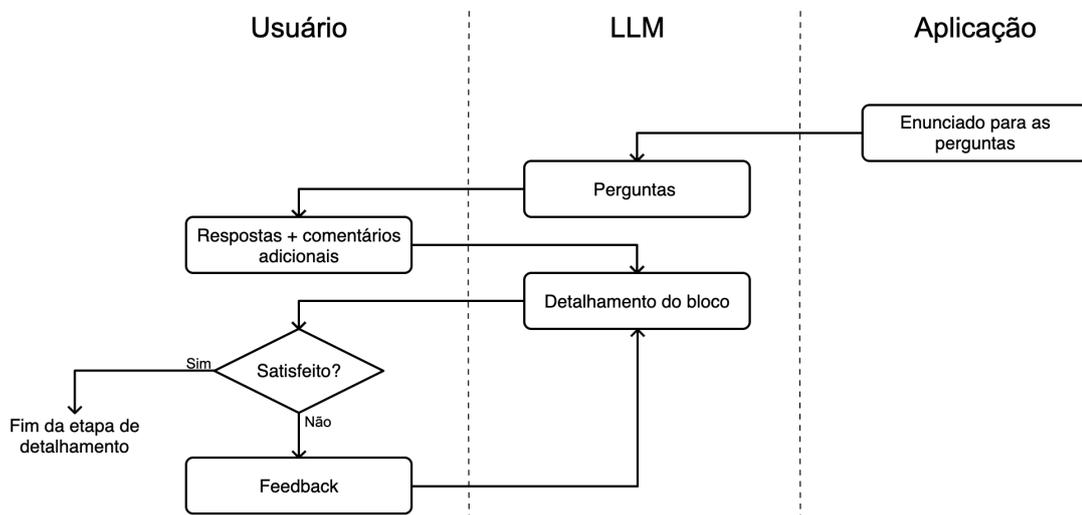
dos sistemas de aquisição de sinais, de forma que o agrupamento do processamento digital é uma limitação planejada.

Para requisitar essa categorização, uma mensagem ao modelo é enviada, sem que se altere o *prompt* de sistema, uma vez que apenas enviar essa mensagem se provou suficiente. Informações referentes à categorização não são expostas ao usuário, pois são internamente processadas para montar os *prompts* da etapa de detalhamento.

3.3.4 Detalhamento dos Blocos

Com a solução em nível de sistema já desenvolvida e a categorização de uso interno já estabelecida, pode-se dar sequência ao detalhamento de cada bloco. O fluxo dessa etapa se assemelha ao da arquitetura, visto que inicia com o modelo fazendo uma sequência de perguntas ao usuário e, com as respostas, fornecendo maiores informações sobre o bloco. A figura 11 expõe as interações do modelo com o usuário e com a ferramenta. Assim como na primeira etapa, a LLM pode fazer até 5 perguntas, porém, afim de evitar cansar o usuário, que precisará fornecer respostas a todos os blocos, é instruído no *prompt* de sistema que se evite perguntas irrelevantes.

Figura 11 – Fluxo de conversa da etapa de detalhamento.



Fonte: o Autor, 2024.

O *prompt* de sistema para a formulação das perguntas é compartilhado entre todas as categorias, visto que é composto apenas por diretrizes e regras gerais, como número máximo de perguntas. Entretanto, a mensagem de usuário que é enviada pela ferramenta solicita que as perguntas sejam específicas à categoria em questão.

Por outro lado, o *prompt* de sistema utilizado para requisitar os detalhes do bloco é muito específico. Esse é o principal mecanismo para a otimização das respostas do GPT, visto que solicitar detalhes com apenas diretrizes gerais - que poderiam ser utilizadas para qualquer parte de um sistema de aquisição de sinais - está fadado a ser genérico.

Assim como no fim da etapa de arquitetura, ao fornecer as sugestões do detalhamento do bloco, é questionada a satisfação do usuário. O fluxo subsequente de *feedback* é o mesmo.

Na abordagem *top-down* adotada neste trabalho, foi estabelecido que os blocos devem ser analisados isoladamente, sem considerar o sistema inteiro para cada decisão. Dessa forma, os blocos devem ser detalhados de forma individual, sem que o modelo saiba o que foi definido para os outros. Portanto, ao iniciar o fluxo de detalhamento de um bloco, independente de quantos já foram analisados anteriormente, o único histórico de conversa disponível ao GPT é o da etapa de arquitetura. O principal motivo por essa implementação é o tamanho do contexto resultante de manter todo histórico de detalhamento. Conforme o número de *tokens* de entrada aumenta, maior a dificuldade do modelo de extrair informações importantes do texto e maior o custo de execução.

A discussão acerca de se essa é a melhor abordagem é válida, visto que ela abre margem para informações conflitantes entre blocos. Entende-se que o mundo analógico depende da interface entre os blocos. Por exemplo, a impedância de entrada de um bloco define os requerimentos da impedância do próximo. No entanto, requisitos críticos que interfaceiam múltiplos blocos podem ser levantados na etapa de arquitetura, para que o detalhamento individual possa estar coerente. Além disso, conforme será apresentado na próxima subseção, referente à etapa de revisão, o modelo terá uma oportunidade no fim do processo de analisar a solução completa e fazer correções, caso necessárias.

Após todos os blocos passarem pelo processo apresentado no fluxograma 11, as mensagens são concatenadas de maneira a formar a conversa completa, contendo a arquitetura e o detalhamento de todos os blocos.

3.3.5 Revisão

Após cada bloco passar pela etapa de detalhamento, a ferramenta chega a última etapa, a de revisão. Nela, o GPT tem a oportunidade de analisar todas as interações que ocorreram ao longo do processo, com exceção dos *loops* de *feedback*, para julgar a necessidade de fazer correções. Visto que é impraticável alterar retroativamente as mensagens, as modificações propostas pelo modelo são simplesmente concatenadas à lista de interações do fluxo de conversa.

Testes preliminares indicaram que, em média, o modelo pode retornar 400 palavras por bloco, o que acarreta em um texto extenso a se ler. Uma solução grande, além de

possivelmente cansar o usuário, é difícil de avaliar. Além disso, contextos muito grandes prejudicam a capacidade dos modelos de destacarem e priorizarem detalhes, em uma espécie de explosão de gradiente. Dessa forma, uma sumarização da solução é necessária.

O resumo é solicitada ao modelo, porém, com rígidas instruções para que se não omitam valores numéricos, cálculos, modelos de componentes, parâmetros ou, de forma geral, qualquer detalhe fornecido ao longo da conversa. Em conclusão, a sumarização serve para remover verbosidade desnecessária, com a qual GPTs adoram encher respostas.

Assim como na etapa de categorização, a revisão e a sumarização não contam com um *prompt* de sistema dedicado, mas apenas com uma mensagem de usuário, enviada pela ferramenta.

3.4 Testes

Ao longo dessa seção, será referenciado o uso de *testbenches*, que são projetos exemplos, cujas soluções são conhecidas e que validarão a qualidade da ferramenta. Contudo, os projetos escolhidos para servirem de *testbenches* serão discutidos apenas na seção 3.5, que adentrará mais nos projetos em si e na sua adaptação ao modelo de *prompts*.

3.4.1 Execuções Automatizadas

A execução de um caso na ferramenta envolve interagir com a interface do modelo, respondendo suas perguntas. Ao fim, será gerada a solução, cuja qualidade pode ser devidamente avaliada a critério do usuário.

No entanto, automatizar execuções para uma ferramenta que utiliza LLMs, na qual as respostas são efetivamente não-determinísticas, é um desafio. Testes automatizados precisam, como o diz o nome, rodar automaticamente, sem a constante necessidade da atenção de um usuário. Tem-se que o grande objetivo do projeto é desenvolver uma ferramenta com a qual pessoas possam projetar sistemas de aquisição de sinais, de forma que é impossível evitar a interação entre aplicativo e cliente.

Portanto, foi necessário desenvolver novos métodos de testes automatizados para avaliar a qualidade da ferramenta, visto que não existem referências nessa área. Dessa forma, entende-se que uma contribuição do projeto é a introdução dessas abordagens de execução e avaliação automatizadas, que está listada como um objetivo específico (ver seção 1.3).

Dois métodos foram desenvolvidos, um que é simples e não faz uso de todas as funcionalidades da ferramenta, e outra que busca realmente emular a interação com um usuário real. A emulação foi pensada no contexto desse trabalho, porém, dados os devidos

ajustes, pode ser utilizada para qualquer aplicação na qual o pilar é a interação com um ou mais usuários.

O primeiro método envia o enunciado do *testbench*² à ferramenta e responde, de forma sistemática, que não possui informação suficiente para responder as perguntas feitas pelo GPT, nem para opinar nas soluções apresentadas. Dessa forma, as únicas informações que o modelo possui para usar ao longo da execução são fornecidas no início. Esse tipo de teste será chamado ao longo do projeto como teste de contexto direto.

Visto que não há como saber como o GPT formulará as perguntas ao usuário, é impossível realizar uma implementação clássica em código para categoricamente respondê-las. Assim, o segundo método procura resolver o problema dessa característica não determinística das LLMs, que impede que o agente inteligente seja removido do processo.

Chamada de teste de contexto investigativo, utilizou-se outro GPT-4-Turbo para cumprir a função de emular o comportamento do usuário. Esse modelo possui um *prompt* de sistema rigidamente construído, cujas instruções o permitem apenas responder perguntas com as informações que estão contidas em uma lista específica. Essa lista é referente ao *testbench* que está sendo executado e é uma simples enumeração de detalhes do projeto que, caso sejam perguntados, podem ser fornecidos à ferramenta. Se o modelo for questionado algo que não está na lista, ele é instruído a dizer que não possui tal informação. O GPT de emulação de usuário é apenas encarregado de responder às perguntas, pois a parte dos *loops* de satisfação sempre será considerada aceitável, de modo a simplificar a execução. Uma Tabela comparativa entre os dois tipos de contextos pode ser vista em 4. Além disso, o *prompt* de sistema parcial fornecido ao modelo está exposto em 3.1. Percebe-se que a lista deve ser concatenada no fim da mensagem, conforme indicado pelo texto:

Tabela 4 – Comparação entre os contextos direto e investigativo.

	Contexto Direto	Contexto Investigativo
Recebe a descrição do projeto	✓	✓
Recebe todos os requisitos no início	✓	
Pode fazer perguntas ao usuário		✓
Utiliza GPT emulador		✓
É avaliado pelo GPT avaliador	✓	✓

Fonte: o Autor, 2024.

Listing 3.1 – Prompt de sistema do GPT emulador.

```
You are impersonating a user that is using a electronics design app.
You will be asked a question and you have to respond it.
You will be provided with a set of available information that you can respond
with.
```

² Cada *testbench* possui um enunciado com o objetivo geral do projeto.

You will only include information in your response that is included in the provided list of available info.

If the list does not contains sufficient information, you must respond by saying "Cannot answer question".

You must follow the exact following markdown template, where all caps words are placeholders that you must fill in.

```
```answer
ANSWER TO THE QUESTION
```
```

Here is the list of information that you have available:

Em resumo, o contexto investigativo exige duas informações: o enunciado do projeto e a lista de respostas disponíveis. Por outro lado, o contexto direto requer apenas o enunciado. Entretanto, para a comparação entre métodos ser justa, ambos devem ter a possibilidade de acesso às mesmas informações. Por conseguinte, o enunciado de contexto direto será composto pela concatenação do enunciado com a lista de informações.

3.4.1.1 Modelo Avaliador

Mesmo com a emulação do usuário no contexto investigativo, nenhum dos métodos propostos consegue avaliar de forma sistemática a solução. O projeto é intrinsecamente qualitativo, visto que o valor real da solução proposta só pode ser atribuído pelo usuário. Contudo, é possível realizar algumas generalizações de modo a considerar que, para a solução ser boa, alguns requisitos mínimos devem ser cumpridos.

Dessa forma, seguindo uma proposta similar ao GPT emulador, foi implementado um terceiro modelo inteligente, que será responsável exclusivamente por avaliar a solução proposta. Para evitar confusões durante o restante do trabalho, o modelo utilizado no teste de contexto investigativo será denominado de GPT emulador. O modelo responsável por propor a solução será chamado de GPT principal e, por fim, o GPT avaliador manterá este nome.

Este receberá o resumo da implementação proposta pela ferramenta, conforme apresentado na seção 3.3.5. Além disso, para cada *testbench*, foram listados os requisitos obrigatórios aos quais a solução deve atender para que essa possa ser considerada ótima. Com estas informações, além de uma série de outras instruções, o modelo avaliador é

encarregado de fornecer a quantidade numérica de requisitos da lista que foram atingidos e um veredito qualitativo. Ambas devem vir acompanhadas de uma justificativa.

Como será visto na seção de *testbenches*, o tamanho da lista pode variar para cada projeto. As justificativas exigidas ao modelo servem apenas como uma ferramenta de *debug*, caso seja necessário retroativamente entender o motivo pela nota ou veredito.

Suponha-se que sejam executadas duas iterações de um projeto de termometria, na qual 1 dos 10 requisitos na lista é que o ganho de um amplificador seja em torno de 50 V/V, enquanto outro requisito seja que a escolha da resistência utilizada para linearização esteja otimizada para 50 °C. Na primeira iteração, o modelo falhou apenas no requisito do ganho, fornecendo um valor de 5000 V/V. Na segunda iteração, o modelo também atingiu 9/10 pontos, mas em decorrência do valor da resistência de linearização estar otimizada para 60 °C. Apesar da nota final de ambas ser a mesma, a primeira iteração nunca poderia ser implementada, visto que o erro percentual do ganho é de 9900%. No entanto, mesmo que parcialmente incorreta, a segunda iteração poderia ser implementada, já que o erro não afetaria tão significativamente o projeto. A solução não é ótima, mas é aceitável.

O problema da simples avaliação quantitativa da solução é que não há como garantir que a qualidade do projeto é sempre diretamente relacionada com a quantidade de requisitos atingidos. Por esse motivo que foi implementado o veredito. As possíveis categorias estão apresentadas abaixo, juntamente com a explicação de cada uma, que é fornecida no *prompt* de sistema, em inglês.

- **Ótima** - caso todos os requisitos forem atingidos;
- **Aceitável** - caso o projeto seja implementável e não possua erros graves, porém nem todos os requisitos foram atingidos;
- **Genérico** - caso o projeto seja implementável, mas está genérico demais para possuir algum valor real;
- **Inviável** - caso o projeto não possua erros teóricos, mas as escolhas feitas impedem que a solução seja viavelmente implementada;
- **Incorreta** - caso o projeto possua erros conceituais, que completamente impedem qualquer tipo de implementação.

O modelo avaliador fornece a nota e o veredito em diferentes interações, pois resultados preliminares indicaram que requisitar tudo de uma única vez aumentava a inconsistência das respostas. Assim, ambos possuem seu próprio *prompt* de sistema, que

podem ser vistos em 3.2 ³ e 3.3. A sequência de caracteres `%s` é utilizada como formatação de *string* e indica onde será inserida a lista de requisitos.

Listing 3.2 – Prompt de sistema do GPT avaliador para a obtenção do veredito.

```
You are a expert project reviewer in the field of analog electronics design.
You have will give a verdict to a electronics project based on these following
requirements:
```

```
%s
```

```
You will categorize the summary into 1 of 5 categories:
```

```
"optimal" - use this if the all requirements were reached
```

```
"acceptable" - use this category if the project can be implemented and doesn't
    have any fatal issues, but the score isn't perfect.
```

```
"generic" - use this category if the project is theoretically correct and can
    be implemented, but too generic to have real value.
```

```
"unfeasible" - use this if the project is theoretically correct, but contains
    elements that makes it impossible to implement
```

```
"incorrect" - the information provided is conceptually wrong
```

```
Requirements marked as essential must be fulfilled in order for a good verdict.
You must think as a project designer if the solution is applicable and well
    thought out.
```

```
You will provide explanations regarding your verdict, the explanations must
    explicitly indicate which requirements were not reached.
```

```
You always follow the exact following template in markdown blocks, where all
    caps words are placeholders for you to fill in.
```

```
Inside the verdict placeholder, you will provide only the chosen category.
```

```
```explanations
EXPLANATIONS HERE
```
```

```
NEWLINE
```

```
```verdict
VERDICT_CATEGORY
```
```

³ Serão mantidos eventuais erros de escrita dos *prompts*, uma vez que não foram identificados antes a sua utilização no projeto.

Listing 3.3 – Prompt de sistema do GPT avaliador para a obtenção da nota.

You are a expert project reviewer in the field of analog electronics design.
You have will evaluate a electronics project based on these following
requirements:

%s

You will indicate which of these items the given project summary successfully
covers.

If, for example, 6 requirements out of 10 are reached, the score must be 6,
regardless of the order in which they appear.

You never provide decimal points.

Sometimes the requirements require you to make some calculations

Sometimes the requirements are fulfilled implicitly

Take into account everything that was said before jumping into conclusions,
not always the information will be clearly said.

If the requirement states that the numeric value is
roughly/loosely/approximately close, then you can be quite generous in
your approximation.

If not, consider a accuracy of 1 percent to evaluate the requirement.

You will provide explanations regarding your score, the explanations must
explicitly indicate the number of the items that were reached and the ones
that weren't.

You always follow the exact following template in markdown blocks, where all
caps words are placeholders for you to fill in.

Inside the score placeholder, you will provide only the numeric integer that
indicates the amount of requirements reached.

```
```explanations
```

```
EXPLANATIONS HERE
```

```
```
```

```
NEWLINE
```

```
```score
```

```
SCORE_NUMBER
```

```
```
```

Do *prompt* destinado ao veredito, destaca-se a instrução a não dar boas classificações a soluções que não atinjam requisitos marcados como essenciais. Dessa forma, mesmo que apenas um não objetivo tenha sido alcançado, o veredito deve ser *inviável* ou *incorreto*.

No *prompt* para obter a nota, explicita-se que o GPT deve considerar uma margem de erro para requisitos numéricos. Observou-se durante alguns testes preliminares que, sem nenhuma diretriz sobre como comparar valores quantitativos, o modelo avaliador exige igualdade absoluta. Por conseguinte, o *prompt* pede uma *accuracy*⁴ de 1% caso não seja explicitamente indicado que o valor deve ser aproximado. Nesse caso, o GPT pode ser ainda mais permissivo.

Sintetizando, a automatização é um conjunto de métodos de emular o usuário, seja utilizando uma LLM, seja por enviar apenas um *prompt* inicial, com um método de avaliação inteligente. O valor numérico resultante permitirá uma análise quantitativa dos resultados e os vereditos permitirão que se avalie qualitativamente a solução. Maiores informações sobre a interpretação dos resultados e do método estatístico serão apresentadas na seção 3.6.1.

3.4.2 Testes Unitários

O objetivo de teste unitário, por definição, é isoladamente testar uma funcionalidade. Para tanto, foram escritos testes unitários para algumas das funções da ferramenta, de forma a garantir que, dentro do escopo de cada uma, a saída gerada, dados certos valores de entrada, estão confiavelmente iguais ao esperado.

Para testes unitários, faz sentido apenas testar os menores componentes do projeto, ou seja, aqueles que não chamam funções dentro de si. Dessa forma, apenas alguns métodos foram cobertos, especialmente aqueles relacionados a geração das imagens utilizando a biblioteca *graphviz* (veja a seção de materiais 3.1), manipulação de texto e interação com o usuário por meio da CLI.

Além disso, foram implementados testes unitários para os modelos GPT utilizados para emular o usuário e para avaliar a solução. Em tese, execuções repetidas de um teste unitário não devem produzir resultados diferentes, porém, no caso não determinístico das LLMs, é impossível garantir que a saída será sempre igual. Contudo, conforme discutido na subseção anterior, os *prompts* de sistema e de usuário que são fornecidos aos GPTs foram formulados com a intenção de tornar as respostas previsíveis. Dessa forma, os testes unitários servem justamente o propósito de fornecer evidências de que - quando o escopo está bem delimitado - os modelos inteligentes conseguem agir suficientemente consistentemente, tais quais funções normais. O modelo GPT principal (que gera a sugestão

⁴ não foi especificado o tipo de *accuracy*, visto que pode ficar a critério do modelo. Caso deseje-se ter essa definição, basta adicioná-la no *prompt*.

de síntese do projeto) não possui um escopo limitado, o que impede a implementação de testes unitários.

No GPT emulador foi providenciado ao modelo a lista de informações disponíveis e as perguntas que estão sendo feitas. Para o modelo avaliador, escreveu-se um sumário de teste e foram dados os requisitos mínimos para que a nota e o veredito pudessem ser atribuídos. Para todos os cenários dos testes unitários envolvendo GPTs, o critério utilizado foi de um erro máximo de 5% para 20 repetições, o que equivale a 1 erro.

Entende-se que testes unitários comprovam a funcionalidade base da ferramenta e, caso qualquer um esteja falhando, não há como prosseguir com as execuções automatizadas.

3.5 Testbenches

Com o intuito de aproximar o trabalho a aplicações reais, foram selecionados 4 projetos para fazerem parte dos testes automatizados. Dessa forma, com a análise a ser feita a respeito dos resultados, é possível julgar com maior confiança se a aplicação possui as competências necessárias para cumprir sua função pretendida de auxiliar projetistas.

Visto que o objetivo do trabalho é mais voltado a parte de sistemas de aquisição de sinais, procurou-se escolher projetos onde os maiores desafios estão no domínio analógico. Todos os projetos incluem, de uma forma ou de outra, etapas digitais. No entanto, não espera-se que a solução adentre em detalhes de implementação em software ou de blocos lógicos, por exemplo.

Dentre os enunciados escolhidos, buscou-se ter diferentes níveis de complexidades e princípios físicos de sensoriamento. Com isso, foi evitado que todos projetos fossem relacionados ao mesmo transdutor, além de evitar que todos possuíssem as mesmas arquiteturas como possíveis soluções.

A complexidade de um projeto é, muitas vezes, subjetiva. No entanto, resultados preliminares indicam uma maior dificuldade de LLMs resolverem problemas extensivamente matemáticos. Dessa forma, alguns dos projetos que serão apresentados nesta seção possuem um caráter mais qualitativo, enquanto que outros exigem que o modelo consiga calcular uma série de valores. Assim, é possível avaliar o desempenho da ferramenta perante diferentes desafios.

As fontes das quais os projetos foram escolhidos são de suma importância para justificar a aplicabilidade do trabalho. Portanto, os projetos foram extraídos de 2 fontes: os laboratórios da cadeira de Instrumentação A da UFRGS, que por sua vez, são baseados no livro de Balbinot e Brusamarello (2010) e; os exemplos do livro *Data Acquisition for Sensor Systems* de Taylor (1997).

A Tabela 5 apresenta um comparativo dos diferentes objetivos de cada *testbench*, além do número de requisitos que cada um possui. As subseções seguintes os projetos serão discutidos em mais detalhes, para que se possa compreender o que se está solicitando à ferramenta. Além disso, serão apresentados os 3 *prompts* associados a cada projeto. Vale relembrar que, conforme apresentado na seção 3.4, para rodar os testes, é necessário um enunciado, uma lista de informações adicionais e uma lista de requisitos. O contexto investigativo utilizará apenas a descrição, porém terá a opção de responder perguntas cujas respostas estão contidas na lista de informações adicionais. Por outro lado, o contexto direto não tem direito a responder perguntas e, conseqüentemente, o enunciado e a lista de informações serão concatenadas de modo a formar o enunciado direto.

Tabela 5 – Comparação entre os *testbenches*.

| | Nº Requisitos | Objetivo |
|-------------------------|----------------------|---------------------------------------|
| Lab. Medição angular | 8 | Tanto quantitativo quanto qualitativo |
| Lab. Termometria | 8 | Qualitativo, sem enfoque em valores |
| Projeto acelerometria | 9 | Cálculos dos valores de componentes |
| Projeto pressão e temp. | 12 | Arquitetura e escolha dos blocos |

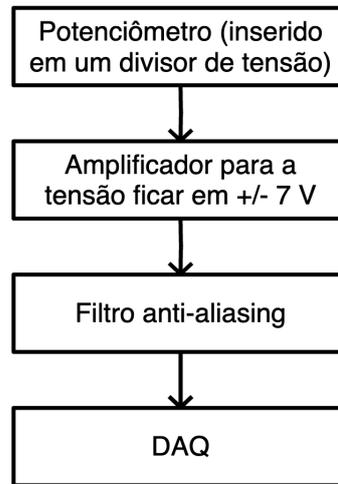
Fonte: o Autor, 2024.

3.5.1 Laboratório de Medição de Posição Angular

Iniciando pelo mais simples, tem-se o projeto de um medidor de posição angular por meio do uso de um potenciômetro. Nesse, considera-se uma estrutura pendular, na qual a extremidade de uma haste está conectada ao eixo do potenciômetro, de forma que a oscilação reflita em uma variação na distribuição de resistência dos terminais do componente. A maior complexidade do projeto está no fato da medição ser feita digitalmente por um DAQ, que exige um filtro *anti-aliasing* e alguma garantia de que a tensão de entrada não será superior ao limite do dispositivo.

Um dos diagramas de blocos viável mais simples que a solução pode assumir está apresentado na Figura 12. Percebe-se que o único condicionamento de sinal imprescindível é o filtro *anti-aliasing*. O objetivo de incluir esse *testbench* é justamente avaliar o desempenho da aplicação nos projetos simples, em vez de apenas nos complexos. Além disso, reconhece-se a possibilidade de soluções corretas, porém desnecessariamente complicadas.

Figura 12 – Uma das possíveis arquiteturas para o projeto de medição de posição angular.



Fonte: o Autor, 2024.

O enunciado, as informações adicionais e os requisitos estão apresentados em sequência.

Listing 3.4 – Enunciado do projeto de medição de posição angular.

I want to develop a project that calculates the angle of a pendulum by using a potentiometer.

The component will be fixed to a wooden structure and another beam will be connected to the adjustable pin.

The analog output will be sent to a DAQ which cannot be altered or tuned.

Use the simplest design possible, which takes into account all requirements.

Listing 3.5 – Informações adicionais do projeto de medição de posição angular.

-
1. The potentiometer is linear
 2. The value of the potentiometer is 10 kOhms
 3. The power source connected to the potentiometer is between -10 and 10 Volts
 4. The angle range that the potentiometer should be able to measure is from 45 to 135 degrees, in which 90 degrees corresponds to the steady position
 5. The sampling rate of the DAQ is 1000 samples per second.
 6. Frequencies around 50 and 60 Hz should be attenuated.
 7. The conditioning should be done in the analog domain
 8. The DAQ is only for acquisition, you can not alter its configuration.
 9. The maximum accepted input voltage for the DAQ is +/- 7 V
-

Listing 3.6 – Requisitos do projeto de medição de posição angular.

1. The potentiometer is used as a voltage divider
 2. The voltage applied to the potentiometer is +/- 10 V
 3. The architecture is simple: there is a voltage divider (with or without a voltage buffer), an anti-aliasing filter and then the DAQ measures the voltage.
 4. The input voltage of the DAQ is centered in 0, for example, +/- 7V.
 5. The maximum voltage applied to the DAQ is 7V (this is absolutely necessary and fatal if false)
 6. There is a low pass filter (or anti-aliasing filter) that avoids a aliasing
 7. There is a filter removing frequencies between around 50 and 60 Hz
 8. The filter has a cutoff frequency and order such that the gain of the signal is of at least -20 dB at 500 Hz
-

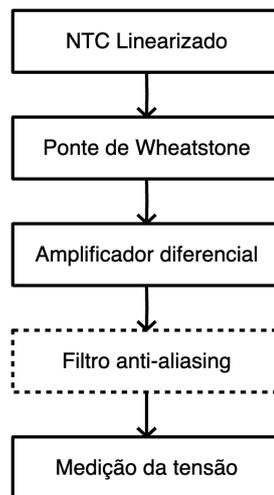
O objetivo do enunciado é fornecer orientações gerais do projeto. Entende-se que apenas essa descrição não é suficiente para obter uma solução viável. Os detalhes críticos que de fato moldam a solução, estão contidos na lista de informações adicionais. Caso as escolhas de projeto sejam corretamente tomadas, mesmo que por diferentes abordagens, todos os requisitos devem ser atingidos. Portanto, não podem existir informações conflitantes nas listas, nem expectativas injustas.

3.5.2 Laboratório de Termometria

Um dos princípios mais simples com o qual se pode trabalhar é o termoresistivo. Neste laboratório, é estudado o comportamento de alguns sensores que se baseiam nesse fenômeno, tanto suficientemente lineares, como o Pt100 da família dos RTDs (*Resistance Temperature Detectors*), quanto os termistores não-lineares NTCs (*Negative Temperature Coefficient*).

Apesar do laboratório ser composto por diversos experimentos, apenas o último deles foi selecionado como *testbench*. Nesse, é pedido que se meça a temperatura da água dentro de um copo de béquer utilizando um NTC. No entanto, em vez de diretamente medir a resistência e correlacioná-la à temperatura, pede-se que seja realizado o condicionamento do sinal de modo que a tensão elétrica possa ser medida com um multímetro comercial. Dada a característica não linear dos NTCs, uma etapa de linearização é essencial, preferencialmente pelo método de três pontos. A Figura 13 representa o diagrama de blocos da arquitetura proposta pelo enunciado do laboratório.

Figura 13 – Possível solução do laboratório de termometria de Instrumentação A



Fonte: o Autor, 2024.

Nota-se que a adição do filtro *anti-aliasing* é opcional, considerando que a medição será feita com um multímetro.

Abaixo estão apresentados os textos que compõem o *testbench*.

Listing 3.7 – Enunciado do projeto de termometria.

I want to develop a project that measures the temperature of water inside a beaker with a thermistor and that outputs a voltage that can be easily measured by a multimeter.

Listing 3.8 – Informações adicionais do projeto de termometria.

-
1. The output voltage range to be measured by the multimeter should be between 0 and 20 Volts.
 2. The temperature that will be measured is between 10 and 90 degrees celsius.
 3. The self heating effect must be considered and must be less than 1 percent.
 4. There should be no ADC other than the one inside the multimeter (which can't be changed).
 5. All conditioning should be done in the analog domain
 6. The sensor to be used is a NTC Vishay NTCLE100E3.
 7. The NTC must be linearized for the midpoint of the input range by a resistor in parallel
-

Listing 3.9 – Requisitos do projeto de termometria.

-
1. The output is being measured by a multimeter
 2. There should be amplification stage and the gain must be provided and justified
-

3. The output voltage range to be measured by the multimeter is between 0 and 20 Volts
 4. The NTC is linearized by a resistor optimised for the midrange 50 C .
 5. The NTC is linearized (this is absolutely necessary and fatal if false)
 6. The architecture should be roughly consisting of the sensor, a linearization stage, amplification, optional filtering and measurement. Or some variation not too far off of this.
 7. The sensor used is an NTC (this is absolutely necessary and fatal if false)
 8. The self heating effect is taken into account and there the maximum current that passes through the NTC is known.
-

Observa-se que na lista de informações adicionais está explicitado o desejo pela ausência de um ADC. Contudo, visto que existe uma conversor embutido no multímetro, adicionou-se esse adendo para não confundir o modelo. Escolheu-se limitar o escopo desse cenário para apenas o domínio analógico para testar a capacidade do GPT de simplesmente aceitar o uso de um multímetro digital, mesmo que essa escolha possa ser considerada inferior.

De forma geral, esse *testbench* possui o objetivo de ser o mais qualitativo dentre os 4. Apesar de existir a possibilidade de fornecer maiores informações do sensor para permitir que o valor do resistor de linearização seja calculado, por exemplo, escolheu-se não fazê-lo. Entende-se que a ferramenta pode auxiliar projetistas de diferentes maneiras e a capacidade de simplesmente fornecer uma visão qualitativa da implementação é válida - desde que esteja correta. Como será visto nos próximos *testbenches*, será avaliado o caráter quantitativo também.

3.5.3 Acelerômetro Portátil para Vibrações de Baixa Frequência

Retirado de Taylor (1997), esse projeto é referente a um acelerômetro portátil para a medição de vibrações de baixa frequência. O livro traz esse exemplo de forma mais vaga e sem uma grande oferta de contextualização e aplicação; no entanto, fornece o passo-a-passo para calcular os requisitos do projeto. Dessa maneira, em oposição ao *testbench* de termometria, esse é mais focado na habilidade do modelo de realizar cálculos matemáticos.

A arquitetura do projeto é composta por: um piezoelétrico seguido de um amplificador de carga e, subsequentemente, de um amplificador de tensão. A aquisição em si do sinal, isso é, o uso de um DAQ, multímetro, entre outros, é deixado a critério do modelo, dada a informação que é para um dispositivo portátil.

No exemplo fornecido pelo livro, o autor desconsidera o uso de um amplificador de tensão e procura conseguir todo o ganho necessário no amplificador de carga. Contudo,

discute-se que essa não seria a melhor abordagem, uma vez que exigiria um valor demasiadamente grande para a resistência de *feedback*, além de amplificadores operacionais FET específicos.

O autor também propõe o uso de um extensômetro em vez de um piezoelétrico, mas decidiu-se restringir o escopo desse *testbench* apenas ao piezoelétrico, visto que o extensômetro já está presente em um dos projetos a serem testados.

As informações que o exemplo providência no seu enunciado estão listadas abaixo:

- Sensibilidade do piezoelétrico: $100 \frac{pC}{g}$ (em que g é a aceleração decorrente da gravidade);
- Oscilação do sinal a ser medido: 2 Hz;
- Amplitude do sinal a ser medido: 5 cm;
- Sinal de saída desejado: 1 V pico a pico;
- Resposta de baixa frequência de -3 dB para 0,25 Hz;
- *Offset* desejado inferior a 10 mV.

A partir dessas informações, deseja-se encontrar o valor da capacitância e resistência de *feedback* do amplificador de carga e máxima corrente de polarização. No entanto, primeiro é necessário encontrar a carga de saída pelo amplificador. Para isso, encontra-se a aceleração pico-a-pico com

$$a_{PP} = (2\pi f)^2 \times x, \quad (10)$$

em que f é a frequência de oscilação da vibração e x é a amplitude. Substituindo os valores, encontra-se uma aceleração de 1,610 g.

Pode-se, então encontrar a carga de saída por meio da equação

$$Q_{PP} = a_{PP} \times S_{piezo} \quad (11)$$

e encontrar 161,0 pC. O exemplo do livro segue com a equação

$$V = \frac{Q_{PP}}{C_f} = 1V, \quad (12)$$

já que é possível encontrar a capacitância de *feedback* assim. No entanto, essa abordagem é a que não considera o amplificador de tensão subsequente, o que acarretará em uma resistência de *feedback* elevada ao aplicar a equação

$$R_f C_f = (2\pi \times f)^{-1} . \quad (13)$$

Nesta, a frequência utilizada é a de corte, fornecida como 0,25 Hz. Assim, o autor conclui que os valores encontrados são de 161,0 pC para a capacitância e 3955 MΩ para a resistência. De modo que, a corrente de polarização deve ser menos que 2,52 pA para manter um *offset* inferior a 10 mV, conforme calculado por

$$I_{bias} = \frac{V_{offset}}{R_f} . \quad (14)$$

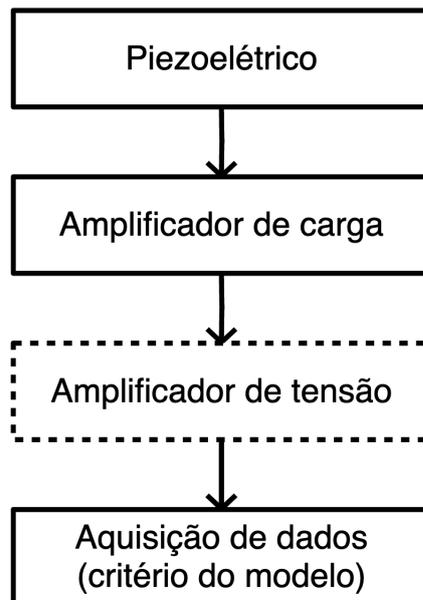
O modelo, por outro lado, é instruído a adicionalmente utilizar um amplificador de tensão e, dessa forma, os ganhos devem devem satisfazer a seguinte equação:

$$V = \frac{Q_{PP}}{C_f} \times G , \quad (15)$$

em que G é o ganho do amplificador de tensão. As equações subsequentes são as mesmas que apresentadas anteriormente.

Portanto, com essas definições é possível compor tanta um diagrama exemplo da arquitetura (ver Figura 14), quanto o enunciado e as listas para esse *testbench*, que estão expostas a seguir.

Figura 14 – Arquitetura sugerida para o projeto de um medidor de vibração portátil.



Fonte: o Autor, 2024.

Listing 3.10 – Enunciado do projeto de acelerometria.

Design a portable device capable of measuring a low-frequency vibration. Make sure to calculate all components and overall values involved in the project, including maximum bias current, charge amplifier components considering a peak-to-peak voltage output.

Listing 3.11 – Informações adicionais do projeto de acelerometria.

1. Use a piezoelectric accelerometer
 2. The accelerometer has a sensitivity of 100 pC/g (g is the acceleration due to gravity)
 3. Use the peak to peak acceleration to calculate charge
 4. An input oscillation has a frequency up to 2 Hz
 5. The input oscillation has a peak amplification of 5 cm.
 6. Output voltage should be 1 V peak to peak
 7. Low frequency response of the system has to be 3 dB down at 0.25 Hz
 8. The offset should be kept as less than 10 mV
-

Listing 3.12 – Requisitos do projeto de acelerometria.

1. On the charge amplifier, the feedback resistance times the feedback capacitance must be roughly around $2/\pi$ (you can allow up to 10 percent difference from this value)
 2. The charge amplifier gain multiplied by $1.61E-12$ and divided by the feedback capacitance ($G \times 1.61p / C_f$) should be roughly around 1 (you can allow up to 10 percent difference from this value)
 3. The peak-to-peak charge output is calculated to be around 161 pC
 4. The bias current must be provided
 5. 0.01 divided by the feedback resistance must equal the bias current
 6. The project uses a charge amplifier to condition the piezoelectric sensor
 7. The project is optimised for an input oscillation of 2 Hz and 5 cm amplitude and has calculations to back it up.
 8. The output voltage is 1 V peak to peak
 9. The offset should be kept below 10 mV
-

Percebe-se que alguns valores numéricos são mantidos em aberto, de forma que o GPT avaliador também precisa realizar alguns cálculos para julgar se o requisito foi atingido.

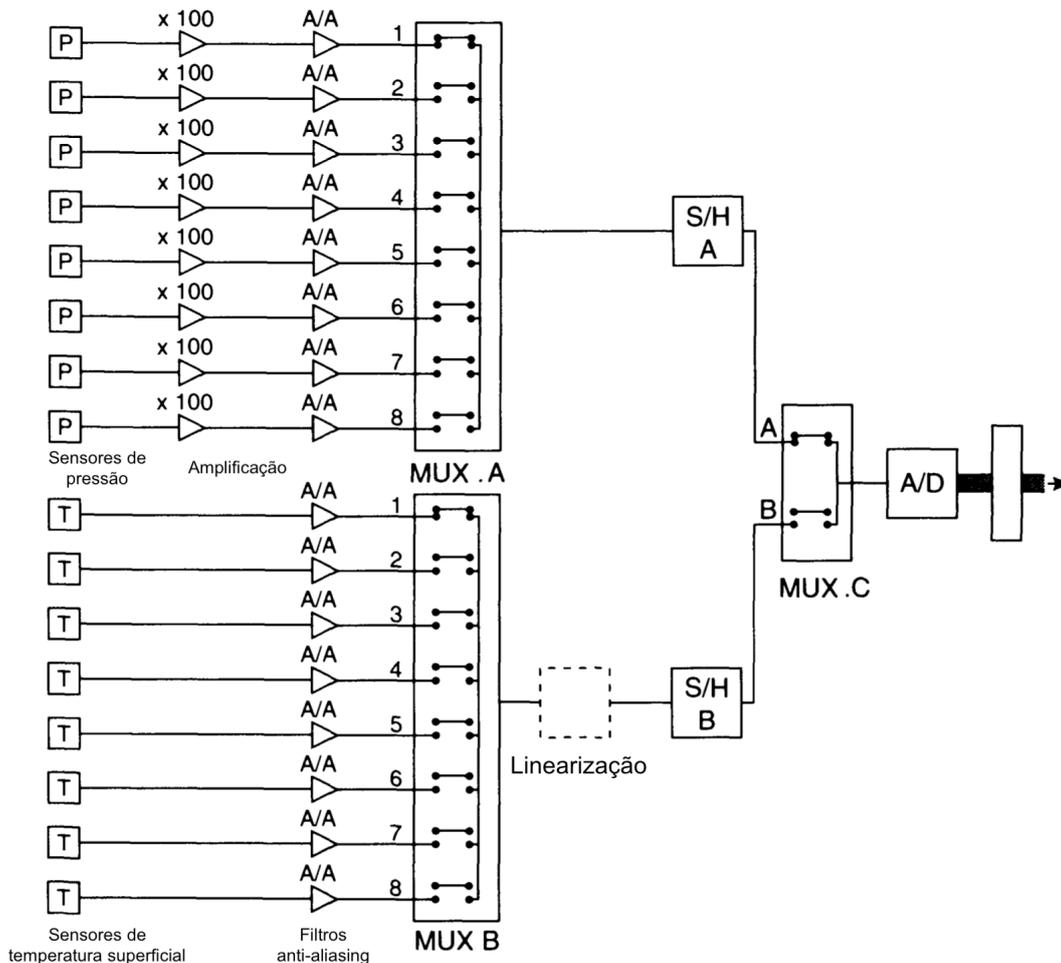
3.5.4 Monitor de Pressão e Temperatura Superficial de Maquinário

Por fim, o último *testbench*, também extraído de Taylor (1997), pede um sistema para adquirir 16 sinais posicionados em diferentes pontos de uma máquina. Oito desses são sensores extensométricos de pressão e os outros são sensores de radiação infravermelha para medir temperatura superficial.

O projeto difere dos outros por ter uma escala maior e por ter dois sensores embutidos no mesmo sistema. Além disso, o autor enfatiza mais a questão de amostragem e multiplexação, sem necessariamente dar instruções concretas para as topologias do condicionamento. No livro, o autor entra nos detalhes referentes à temporização dos sinais, porém, tendo em mente que isso foge do escopo da ferramenta, essa parte foi omitida do *testbench*.

A arquitetura proposta por Taylor pode ser observada na Figura 15. Os *prompts* podem ser vistos a seguir, no mesmo formato que os anteriores.

Figura 15 – Arquitetura sugerida do exemplo do livro *Data Acquisition Systems*.



Fonte: adaptado de Taylor (1997)

Listing 3.13 – Enunciado do projeto de medição de pressão e temperatura superficial.

Design a system to monitor pressure and surface temperature variations at eight points of a machine and the data is to be analysed by computer. Design the acquisition system but don't worry about any software implementations or logic analysis.

Listing 3.14 – Informações adicionais do projeto de medição de pressão e temperatura superficial.

1. The accuracy of both sensors should be at of 1 percent
 2. The pressure sensors have a maximum output of 1 mV
 3. Temperature sensors are radiation detectors having a maximum output of 100 mV
 4. Temperature sensors have a nonlinear scale
 5. Strain-gauge should be used as the pressure sensor
 6. Infrared radiation detectors should be used for the surface temperature sensor.
 7. Assume all pressure sensors have identical sensitivity
 8. Signal frequencies up to 400 Hz are expected
 9. Frequencies higher than 400 Hz (if present) need not be recorded.
 10. The ADC should sample one channel at a time
-

Listing 3.15 – Requisitos do projeto de medição de pressão e temperatura superficial.

1. Both sensors have d.c. output, so no demodulation should be used. (this is absolutely necessary and fatal if false)
2. Both sensors must be amplified (this is absolutely necessary and fatal if false)
3. The pressure sensor must be inserted in a wheatstone bridge and amplified by a instrumentation amplifier
4. An ADC should be used (this is absolutely necessary and fatal if false)
5. Infrared radiation sensors are being linearized either digitally in the computer or by using diode networks or log amplifiers. (this is absolutely necessary and fatal if false)
6. The solution mentions the sampling order strategy, if it'll be sequential, simultaneously, etc
7. The sampling frequency of the ADC is not less than 800 Hz (this is absolutely necessary and fatal if false)
8. The anti-aliasing filter has a cutoff frequency and order such that the gain of the signal is of at least -20 dB at half the sampling frequency
9. The low pass cutoff frequency is higher than 400 Hz and lower than half the total sampling frequency.

10. There are low-pass filters (or anti-aliasing filters) to reduce aliasing, which are positioned before the multiplexer(s)
 11. The project uses multiplexer(s) to choose channels (this is absolutely necessary and fatal if false)
 12. The multiplexer(s) are solid state
-

O enunciado formulado para esse *testbench* é muito resumido, exigindo com que o contexto investigativo faça as perguntas certas, caso contrário, os detalhes não serão repassados à ferramenta. Outro ponto a se destacar é a quantidade de requisitos ditos fatais, que impediriam um veredito positivo caso não atingidos. Isso se deve ao fato da maioria deles serem relacionados ao fluxo obrigatório dos sinais, como amplificação, multiplexação, entre outros. Nem todas essas diretrizes estão explicitamente indicadas no enunciado e nem na lista de informações adicionais, porém são suficientemente intuitivas, uma vez que a habilidade de tomada de decisões do modelo é o que está sendo testado.

3.6 Análise dos Resultados

3.6.1 Visão geral

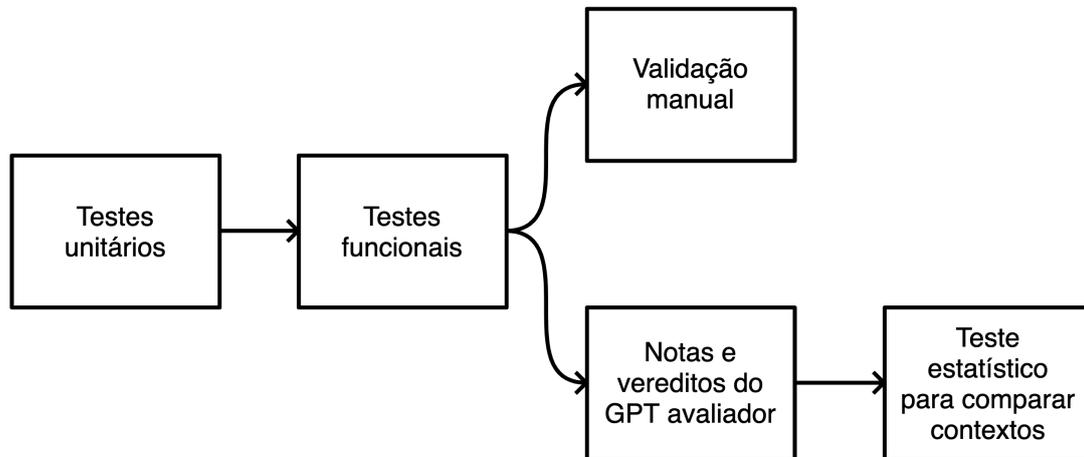
Definidos os *testbenches*, assim como o modelo de testes automatizados, pode-se estabelecer o método com a qual os resultados serão interpretados e avaliados. Como já abordado em alguns momentos do trabalho, o projeto desenvolvido possui características intrinsecamente qualitativas. Mesmo com a definição da *nota*, que indica quantos requisitos a solução atingiu, o uso de métodos estatísticos não é inteiramente possível.

A proposta principal do trabalho, apresentada na seção 1.3, é testar se os modelos GPTs possuem a capacidade de auxiliar o projeto de sistemas de aquisição de sinais. No entanto, a lista de requisitos, o método automatizado de teste e o projeto desenvolvido são todas contribuições do presente trabalho. Nesse sentido, discute-se que não existem bases referenciais externas suficientes para que se possa testar essa hipótese.

Entende-se que é uma possibilidade estabelecer um limiar e aplicar um método estatístico de 1 amostra. Contudo, visto que o limiar também teria de ser definido sem referência ou amostra externa, perde-se valor em ir por esse caminho. O problema continua o mesmo: o estudo não possui base comparativa suficiente para estatisticamente avaliar o desempenho da ferramenta em si.

A Figura 16 apresenta o fluxo de testes que será abordado nesta seção. Percebe-se a bifurcação indica que houveram tanto testes e análises provenientes do GPT avaliador, quanto validações manuais. Os testes unitários, discutidos na seção 3.4, testam o funcionamento base da aplicação e, por isso, estão no início do fluxo.

Figura 16 – Fluxo de testes utilizados para avaliar a ferramenta e os testes automatizados.



Fonte: o Autor, 2024.

3.6.2 Execuções dos Testes Automatizados

Conforme visto na revisão bibliográfica da API da OpenAI (ver seção 2.1.3.4), o número de parâmetros que podem ser alterados em prol de uma variação na resposta é grande. Dentre eles, apenas alguns são de fato interessantes para a ferramenta, visto que não alteram o comportamento de forma artificial. Parâmetros de *bias*, em que se poderia favorecer o uso da palavra *sensor*, por exemplo, acabam por criar uma espécie de *over-fitting*. A temperatura e *top_p* são bons exemplos do que poderia ser parametrizado, pois alteram apenas o determinismo geral do modelo.

Entretanto, nessa parte do trabalho encontrou-se uma limitação de projeto: o custo. O fluxo de conversa é formado por várias etapas, de modo que o preço cobrado por sua execução torna-se considerável. Em média, para cada execução completa da ferramenta, o valor cobrado está entre 0,50 USD e 1,00 USD ⁵. Com quatro *testbenches* e duas formas de contexto (direto e investigativo), tem-se já 8 grupos amostrais. Considerou-se que, para obter uma robustez estatística mínima, são necessárias 20 iterações para cada combinação de *testbench* com contexto, de forma a totalizar 160 iterações. Qualquer variação de outros parâmetros acarretaria em um custo adicional significativo.

Portanto, foi decidido definir a temperatura por meio de testes preliminares. Conforme apresentado no próprio manual de referência da OpenAI, não é recomendado variar tanto *top_p* quanto temperatura.

Surpreendentemente, os testes preliminares indicam que temperaturas maiores - por volta de 0,8 - apresentam resultados melhores. Geralmente, esse parâmetro é associado

⁵ Valores altamente sujeitos a variações ao longo do tempo. Confirme em OpenAI (2023d) se a precificação se mantém a mesma.

com a *criatividade* do modelo, e artigos discutem que tarefas mais voltadas a área das exatas se beneficiam de menores temperaturas. No entanto, entende-se que a diagramação e o detalhamento de projetos de aquisição de sinais exige competências além do domínio matemático.

Em conclusão, houveram 20 execuções para cada *testbench* e para ambos contextos, totalizando 160 iterações. Todas essas estarão com a temperatura fixa em 0,8 e o *top_p* com o valor padrão.

3.6.3 Validação Manual

Uma vez que o presente trabalho não está propondo apenas a ferramenta em si, mas também o método de teste e de avaliação, uma etapa de validação manual é necessária para analisar o desempenho de todas as partes em conjunto. Do modo que o projeto foi desenvolvido, os arquivos gerados ao fim da execução contém tanto a arquitetura e o detalhamento da sugestão construída pelo GPT, quanto as notas e os vereditos, cada uma com suas respectivas justificativas. Assim, é possível julgar a qualidade da solução e do modelo avaliador.

Poderia-se validar a solução em mais detalhe caso as trocas de mensagens com os modelos fossem individualmente analisadas, uma vez que estas são salvas. Contudo, por questões logísticas, não é viável fazer isso para cada uma das 160 iterações. Dessa maneira, serão utilizados apenas os arquivos principais para a validação, porém, em casos que se exija uma investigação mais aprofundada, serão utilizados todos os *logs* disponíveis.

Por meio dessa verificação manual, espera-se encontrar, principalmente, inconsistências na avaliação do GPT avaliador, uma vez que as notas e os vereditos podem conter falsos positivos e falsos negativos. Assim, será possível tirar conclusões a respeito do desempenho da ferramenta com maior embasamento.

3.6.4 Análise e Exposição dos Dados

Em vista dos pontos apresentados, optou-se por dois conjuntos de análises para esse trabalho. O primeiro será referente aos resultados em si, sem que haja a aplicação de algum método estatístico. As notas, assim como os vereditos, serão exibidos e discutidos, de modo que se possa testar a hipótese apresentada na seção 1.3, de objetivos. Será calculada a média para cada um dos diferentes grupos a ser testado, assim como apresentados os melhores e piores resultados. Destaques encontrados nas validações manuais, tanto positivos, quanto negativos, também serão expostos - assim como haverá uma discussão referente as suas possíveis causas.

No entanto, para testar se um tipo de contexto (direto ou investigativo) teve um desempenho significativamente melhor ou pior que outro, será aplicado o *Mann-Whitney*

U-Test. Apresentado na seção 2.4, esse método estatístico permite testar a hipótese de que um grupo amostral tem dados significativamente diferentes de outro. No caso anterior em que se deseja testar se a ferramenta como um todo desempenha bem, a comparação entre tipos de contexto é possível pois não depende de referências externas, como um limiar ou uma base comparativa.

Como os *testbenches* possuem objetivos distintos e procuram explorar diferentes capacidades do GPT, não serão comparados estatisticamente. Será possível expositivamente analisar e discutir o desempenho de cada execução, mas não há valor em comprovar por meio do *U-Test* se um *testbench* foi melhor que o outro.

3.7 Aplicação CLI

O trabalho propõe o desenvolvimento de uma ferramenta capaz de auxiliar projetistas. No entanto, visto que o motor por trás da ferramenta é uma série de mensagens pré-formuladas a serem enviadas às LLMs, é possível discutir a probabilidade de apenas enviar essas mensagens diretamente, sem a necessidade de uma camada de aplicação acima. Entretanto, a quantidade de *prompts* e de validações intermediárias realizadas durante o fluxo de conversa proposto somente se tornam praticáveis se feitas por meio de uma ferramenta dedicada.

Portanto, para tornar a ferramenta um aplicativo de fato, foi desenvolvida uma aplicação CLI (sigla do inglês *Command Line Interface*, que se traduz para "Interface de linha de comando"). Dessa forma, o usuário não precisa realizar manipulação alguma dos *prompts* de sistema, das respostas do GPT e da comunicação HTTP, precisando apenas responder perguntas.

3.7.1 Comandos

A ferramenta conta com uma pequena quantidade de comandos, que servem para o propósito de iniciar, limpar e testar o projeto. Os comandos são todos do formato *Make*, e se encontram no arquivo *Makefile* na raiz do projeto.

1. **make build** compila o projeto real em um executável chamado *chattronics*;
2. **make run** executa o binário *chattronics*;
3. **make build-direct** compila o projeto de teste automatizado de contexto direto em um executável chamado *chattronics-direct*;
4. **make run-direct** executa o binário *chattronics-direct*;

5. **make build-investigative** compila o projeto de teste automatizado de contexto investigativo em um executável chamado *chattronics-direct*;
6. **make run-investigative** executa o binário *chattronics-investigative*;
7. **make clean** deleta os binários compilados;
8. **make unit-tests** roda todos os testes unitários do projeto;
9. **make delete-all-runs** deleta a pasta *runs* contendo todos os *logs* e resultados de execuções passadas.

O comando de *run* leva dois argumentos: modelo e temperatura. Ambas configurações são globais, ou seja, afetam todas as interações feitas com o modelo. O exemplo 3.16 mostra como o comando *make build* e *make run* devem ser chamados caso deseje-se utilizar o modelo GPT-3.5-Turbo e uma temperatura de 0,7. Os valores padrões de modelo e temperatura utilizados, caso não sejam especificados, são, respectivamente, gpt-4-turbo e 0,8.

Listing 3.16 – Exemplo de uso do comando *make build* e *make run*

```
make build
make run model=gpt-3.5-turbo temperature=0.7
```

Os comandos *run-direct* e *run-investigative* também levam ambos parâmetros mencionados acima, mas com a adição de mais dois: número de iterações e *testbench*. O *testbench* é o que indica à ferramenta quais *prompts* e enunciados utilizar nos testes, apresentados na seção 3.5.

3.7.2 Experiência do Usuário

Visto que ferramenta é uma aplicação CLI, toda interação do usuário é feita através da linha de comando. Todas as mensagens exibidas na tela possuem cores específicas para diferenciar a fonte ou importância da mensagem. Mensagens ciano são vindas da ferramenta em si e mensagens verdes são mensagens provenientes da LLM. Texto em amarelo representa avisos importantes ao usuário, como, por exemplo, instruções sobre como utilizar a ferramenta. Mensagens em vermelho indicam que houve algum erro durante o processo ou que o usuário enviou incorretamente alguma mensagem.

A respeito do formato cujo usuário deve inserir o texto, é necessário sempre digitar *enter* duas vezes para enviar a mensagem. Isso foi implementado para que se possa escrever textos com mais de uma linha, sem que se envie acidentalmente uma mensagem incompleta. Além disso, para não causar nenhum erro, é importante que a mensagem não esteja vazia.

Durante a etapa de arquitetura, na qual o diagrama de blocos é gerada, a imagem PNG irá automaticamente abrir e ser apresentada ao usuário. Nota-se que, por limitações da própria linha de comando, não é possível exibir as imagens de forma *inline*, isto é, embutidas do texto. Ao fim, será providenciado um link com a localização na máquina local para a pasta contendo a imagem e o resumo.

3.7.3 Logs e Histórico de Execuções

A cada execução da ferramenta, uma pasta nova é criada seguindo o formato de nome *TIPO_Mar16_14-57-30*, em que, nesse exemplo, a data de execução seria no dia 16 de Março às 14 horas, 57 minutos e 30 segundos. O *tipo* indica a origem da execução, que pode ser um teste unitário, uma execução de contexto direto ou investigativo, ou uma execução real. Dentro dessa pasta, encontra-se um arquivo chamado *console.logs*, que contém, em formato de texto, todos os registros que ocorreram ao longo do processo. Existem diferentes níveis para esses avisos, sendo eles: *INFO*, para simples informações; *DEBUG*, para maior granulação e verbosidade; *WARN*, para avisos de potenciais erros; e *ERROR*, para erros.

Além desse arquivo, a pasta conta com um número variável de outras pastas, cujos nomes são simples enumerações começando em zero. Como os testes automatizados são executados repetidamente, cada pasta numerada indica uma dessas iterações. Dentro, é armazenada uma pasta de *gpt_logs*. Os *logs* contidos nessa pasta, diferentemente daquela mencionada anteriormente, armazenam as conversas com os GPTs em formato JSON. O número de arquivos presente nessa pasta será 1 ou 3, dependendo do tipo de execução. Sempre haverá um arquivo chamado *engineer.json* - referente ao GPT responsável pela solução em si, porém, se for um teste, haverá um arquivo chamado *mock_user.json* - do modelo emulador de usuário, e um arquivo *evaluator.json* - do modelo avaliador. Um exemplo do formato JSON existente nesses arquivos está apresentado abaixo.

Listing 3.17 – Exemplo de *log* das interações com o modelo.

```
1 [
2   {
3     "prompt": [
4       {
5         "role": "system",
6         "content": "Example system message"
7       },
8       {
9         "role": "user",
10        "content": "Example user message"
11      }
12    ]
13  }
```

```
12 ],
13   "response": "Example model response",
14   "input_tokens": 169,
15   "output_tokens": 18,
16   "cost": 0.0014381830215,
17   "finish_reason": "stop"
18 }
19 ]
```

4 Resultados

O capítulo de resultados está dividida em 6 seções e inicia com uma breve discussão a respeito do funcionamento da ferramenta. Uma vez que os testes unitários testam a funcionalidade base da aplicação, esses também serão abordados nessa mesma seção.

Em seguida, serão expostas as melhores soluções obtidas para cada *testbench*, acompanhadas de uma breve discussão a respeito do porquê das escolhas. O veredito e a nota, fornecidos pelo GPT avaliador, não foram critérios para essas classificações, visto que apenas a validação manual foi considerada. O objetivo dessa exposição parcial - prévia à análise completa - é que o leitor compreenda o formato da solução gerada pela ferramenta. Nas seções seguintes, não serão mais apresentadas as soluções completas, apenas serão destacadas algumas partes relevantes à discussão.

Na terceira seção, serão expostas as tabelas contendo as notas, vereditos e alguns dados de cada um dos *testbenches* e contextos (direto e investigativo). A validação manual não será feita nessa etapa, de modo que os resultados indicam apenas as considerações do GPT avaliador.

Na seção subsequente haverá uma discussão dos resultados obtidos em cada *testbench*, tendo em vista as validações manuais. Dessa forma, será analisada não somente o desempenho da ferramenta, mas também do modelo avaliador. Aqui, serão apresentadas algumas inconsistências e erros presentes nos diagramas e nos detalhes escritos.

Por fim, serão feitos os testes estatísticos para determinar se a hipótese de que o contexto direto e o investigativo apresentam resultados significativamente diferentes. Será, também, nessa seção, que se discutirá a validade dessa análise, tendo em vista que as notas são provenientes de um modelo inteligente que possui suas limitações.

4.1 Funcionamento da Ferramenta

4.1.1 Testes Unitários

Como apresentado na seção de metodologia, tanto funções estáticas, quanto funções dependentes do GPT foram testadas unitariamente. Visto que estes testes atuam diretamente no funcionamento isolado da função, garantiu-se por meio desses que os componentes estavam de acordo com o esperado. Isso não significa, no entanto, que as respostas da ferramenta como um todo estavam de acordo, uma vez que esse não é o objetivo por trás dos testes unitários. O projeto desenvolvido contou com uma série de

pacotes ¹, arquivos e funções, e os testes unitários foram imprescindíveis para validações intermediárias, removendo a limitação de apenas conseguir testar o funcionamento lógico da aplicação por meio de execuções completas.

Caso algum teste não esteja funcionando, existe uma brecha evidente na funcionalidade do projeto, considerando que o cenário descrito no teste está conforme o esperado. Portanto, antes de iniciar as execuções, confirmou-se que a taxa de acerto dos testes fossem de 100%.

4.1.2 Erros Obtidos

Apesar da garantia dos testes unitários, estes apenas validam os cenários que foram implementados, de modo que, se algum *edge case*² não for coberto, podem existir erros durante a execução. Ao rodar os testes para o trabalho, houveram algumas instâncias nas quais *edge cases* impediram a continuação da iteração. Para as 160 iterações, isso ocorreu um total de 6 vezes, ou seja, por volta de 3,75%, de modo que é possível utilizar a definição de caso extremo.

Contudo, esse número ainda pode ser considerado alto e, mesmo que os erros ocorram, o ideal é que exista alguma estratégia de resiliência implementada para impedir o término da iteração. Embora estas tenham sido interrompidas, assim perdendo toda solução construída até o momento, o pior cenário é aquele que derruba a ferramenta inteira, e não apenas uma das n iterações. Por conseguinte, os erros, apesar de existirem, não foram suficientes para parar de vez a execução. Além disso, nas situações em que ocorreram problemas, iterações adicionais foram feitas de modo a garantir que existissem 20 para cada grupo amostral.

Em relação aos testes unitários implementados para as funções dos GPTs emulador e avaliador, garantiu-se, dentro dos cenários desenvolvidos, que o resultado esperado fosse atingido para mais de 95% das execuções, porém, como poderá ser observado nas subseções seguintes, a característica não-determinística dos modelos é dominante, de modo que não se pode ter certeza do seu comportamento, mesmo com as medidas tomadas.

4.2 Melhores Resultados por *Testbench*

Nessa seção serão expostas as figuras contendo a solução textual gerada, adaptada ao formato visual (sem alteração do conteúdo), as quais serão brevemente discutidas. Uma vez que a quantidade de texto fornecido pelos modelos GPTs é grande, os resumos originais

¹ se refere ao modo no qual dependências são organizadas em *Go*. Estes ditam o contexto no qual diferentes arquivos estão inseridos, de modo que componentes privados só podem ser acessados por outros pertencentes ao mesmo pacote, por exemplo.

² nome dado à cenários incomuns, que acontecem apenas dadas circunstâncias específicas. Uma possível tradução do inglês é *caso extremo*

das melhores soluções foram colocadas no apêndice B, assim como as notas, vereditos e justificativas gerados pelo modelo avaliador.

4.2.1 Laboratório de Medição de Posição Angular

Ao observar o diagrama de blocos na Figura 17, percebe-se que, salvo a escolha do *buffer*, a arquitetura segue o formato proposto na seção de metodologia. Em vez de utilizar um único filtro passa baixas, que seria uma abordagem mais simples, a escolha de um *Notch* e de um *anti-aliasing* também era possível. Na seção de validação manual (ver 4.4), será discutido como o GPT possui dificuldade de implicitamente cumprir um requisito. A menção de frequências de 50 e 60 Hz já é suficiente para quase que forçar o GPT a implementar blocos que diretamente atuam nessa restrição. Nota-se, também, que a ferramenta chamou o filtro *Notch* de passa-bandas, quando na verdade é um rejeita-faixas.

Apesar da tensão nos terminais do potenciômetro ser de -10 e +10 V, a faixa de operação do potenciômetro é de 90 graus, de modo que a ferramenta está correta ao considerar que a entrada de tensão do estágio amplificador será de ± 5 V. Esse resultado indica uma concordância entre os blocos de sensoriamento e de amplificação, que, como será visto em seções seguintes, nem sempre ocorre. Além disso, o resumo sugere que diodos sejam utilizados antes do DAQ para proteger os dispositivos de tensões superiores ao seu limite de 7 V, que agrega à solução. Contudo, uma vez que a ferramenta optou por um amplificador não-inversor, que possui alta impedância de entrada, a escolha do *buffer* pode ser considerada desnecessária.

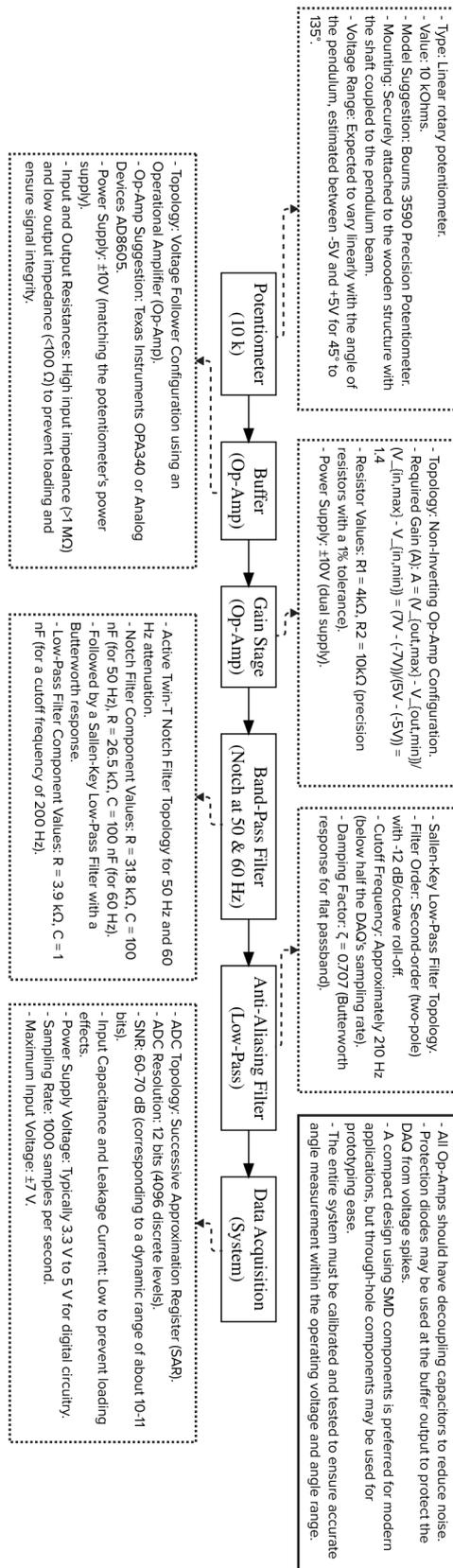
Apesar de algumas das escolhas não serem necessárias, como o *Notch* e o *buffer*, essa solução não cometeu erros teóricos, como equações ou implementações incorretas. Assim, dentre as 40 iterações do laboratório de posição angular, essa foi considerada a melhor execução.

4.2.2 Laboratório de Termometria

A Figura 18 apresenta uma arquitetura coerente com o esperado, visto que as etapas de linearização, ponte, amplificação diferencial e passa-baixas estão todas incluídas. Uma melhoria ao diagrama seria o uso de setas duplas para indicar a saída da ponte de *Wheatstone*, que é uma tensão diferencial.

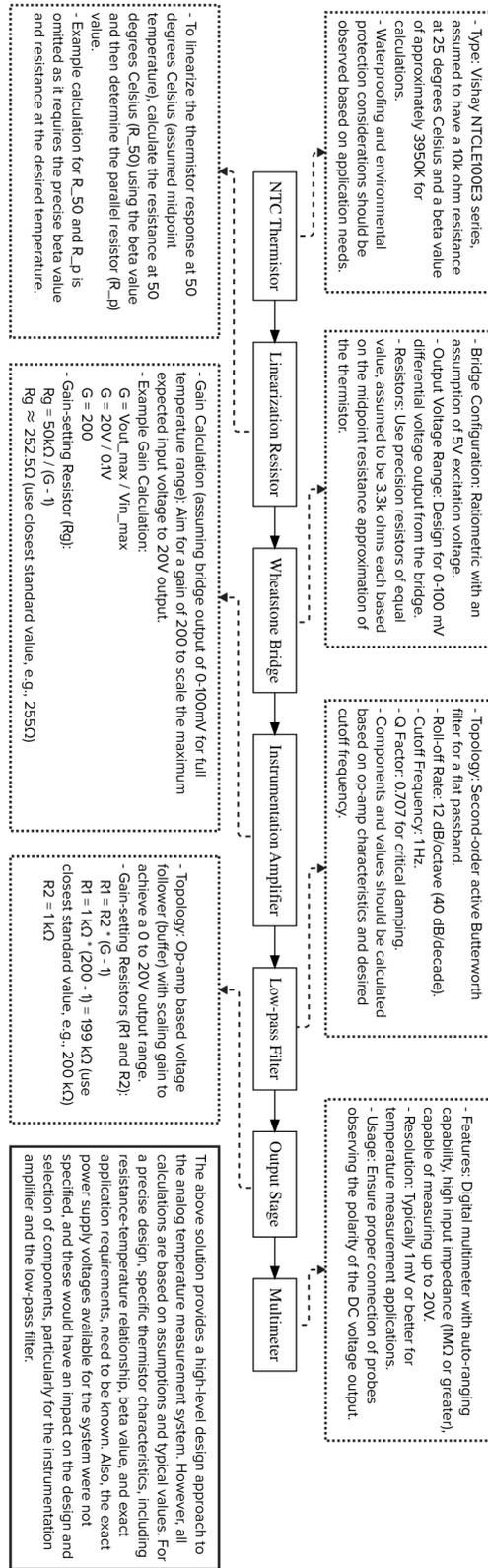
Esta foi considerada a melhor iteração pelo fato do diagrama estar coerente e pelo resumo conter informações viáveis de implementação. Não apenas o ganho do amplificador foi calculado, mas também o valor dos resistores da ponte. Sabe-se, de outras iterações, que o modelo possui a capacidade de informar valores corretos de filtros passa-baixas, portanto, um dos pontos negativos dessa solução foi a ausência disso.

Figura 17 – Diagrama do melhor resultado para o laboratório de medição de posição angular, obtida da iteração 7 com contexto direto.



Fonte: o Autor, 2024.

Figura 18 – Diagrama do melhor resultado para o laboratório de termometria, obtida da iteração 3 com contexto investigativo.



Fonte: o Autor, 2024.

Outro problema foi a inclusão de um ganho de tensão no bloco denominado *Output Stage* (estágio de saída, no inglês). No entanto, atribui-se a esse erro, também, a falta de concordância entre os blocos, uma vez que a arquitetura não indicava que esse bloco deveria possuir ganho, mas o detalhamento optou por tal de qualquer forma. Entende-se ao ler a implementação proposta para o estágio do amplificador de instrumentação, que todo o ganho do projeto está dimensionado nesse único bloco.

4.2.3 Projeto de Acelerometria

A Figura 19 é referente ao melhor resultado para o projeto de acelerometria, em que o *Analog Front End* apresenta os blocos esperados para esse *testbench*. Percebe-se que o bloco de *anti-aliasing* e de microcontrolador não estão especificados no sumário. Acredita-se que essa omissão pode ser dada pela quantidade de informação no projeto, de modo que o modelo tenha ignorado esses blocos.

A parte de processamento digital da arquitetura possui o erro de posicionar o filtro *anti-aliasing* após o ADC, que é suficiente para inviabilizar o projeto. Entretanto, visto que o foco desse *testbench* não era a aquisição e, sim, os cálculos dos valores dos componentes do condicionamento, esse critério não foi o suficiente para desconsiderar a solução como sendo a melhor entre as outras.

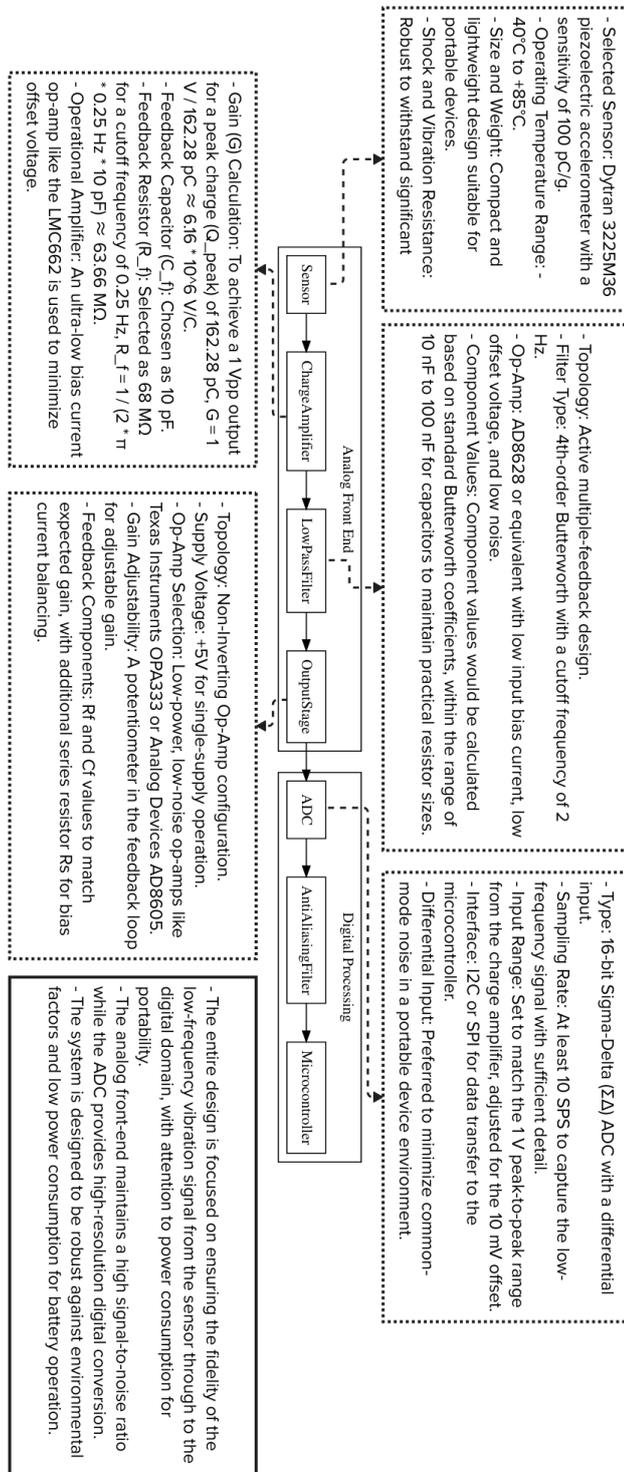
Como será observado nas seções seguintes, os modelos GPT tiveram dificuldade em corretamente calcular os valores dos componentes, portanto, o diferencial dessa execução foi conseguir dimensionar um ganho pico a pico de 1 V, e conseguir encontrar o valor esperado da carga de saída do amplificador de carga. Contudo, essa iteração não foi isenta de erros matemáticos, pois não conseguiu calcular valores coerentes para satisfazer a frequência de 0,25 Hz do amplificador de carga.

Além disso, será posteriormente discutido como os resultados do projeto de acelerometria foram prejudicados pela falta de comunicação entre os blocos, ou seja, pelo modelo não ter acesso ao histórico de outros detalhamentos. No entanto, em comparação às outras execuções, considera-se que essa tenha sido a melhor iteração para o projeto de acelerometria.

4.2.4 Projeto de Pressão e Temperatura Superficial

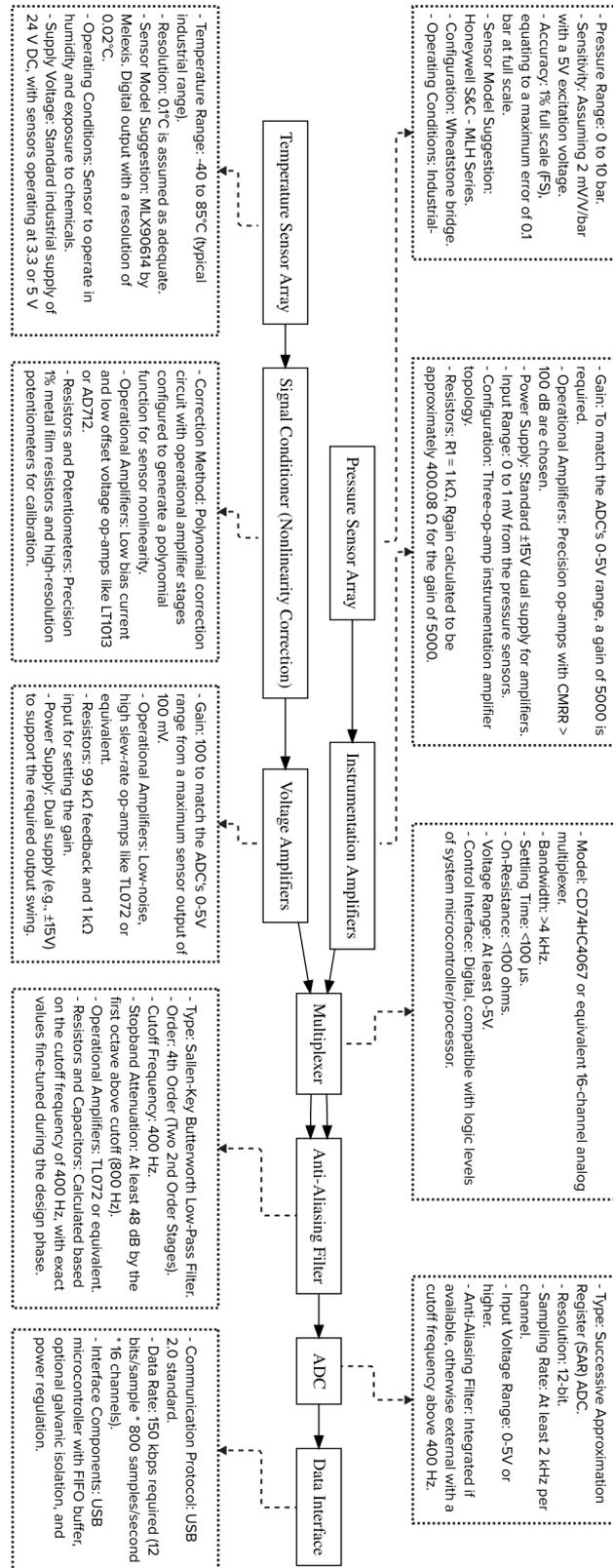
A Figura 20 apresenta o diagrama de blocos do projeto de medição de pressão e temperatura superficial. Percebe-se a seta dupla, que não deveria estar presente na saída do multiplexador. O resumo especifica o uso de uma ponte de Wheatstone para os extensômetros, mas isso deveria estar explicitado no diagrama também.

Figura 19 – Diagrama do melhor resultado para o projeto de acelerometria, obtida da iteração 18 com contexto direto.



Fonte: o Autor, 2024.

Figura 20 – Diagrama do melhor resultado para o projeto de medição de pressão e temperatura superficial, obtida da iteração 8 com contexto direto.



Fonte: o Autor, 2024.

Essa execução fornece tanto os valores dos ganhos, quanto da frequência de corte do filtro *anti-aliasing*, sem que sejam feitos erros teóricos. O requisito que não foi cumprido foi a do filtro estar posicionado antes do MUX.

4.3 Resultados de Testes com Notas e Vereditos do Modelo Avaliador

Nessa seção serão apresentados apenas as notas e os vereditos de cada iteração, para ambos contextos de cada *testbench*. A partir dos dados obtidos poderão ser analisadas o desempenho relativo da ferramenta, ou seja, o desempenho a partir dos critérios do GPT avaliador. Em decorrência da proposta do trabalho, os dados trazem informações de toda pilha tecnológica implementada, que abrange tanto a parte da ferramenta em si, quanto das metodologias de testes e de avaliação. Ao apresentar apenas as notas e os vereditos é possível contextualizar o que será posteriormente observado com a validação.

4.3.1 Laboratório de Medição de Posição Angular

O *testbench* relacionado ao potenciômetro obteve, em sua maioria, vereditos positivos e notas próximas ao máximo de 8, como indicam as Tabelas 6 e 7. Contudo, houveram 6 instâncias nas quais o veredito não foi positivo, ou seja, não foi nem *ótimo* nem *aceitável*. Analisando o motivo pelo qual esses casos ocorreram, percebe-se que algum dos requisitos ditos como obrigatórios não foi julgado como atingido, mais especificamente, a tensão máxima fornecida ao DAQ não foi claramente limitada a $\pm 7 V$. A Tabela 8 apresenta as médias e os desvios padrão, além de possuir uma contagem dos vereditos obtidos.

Tabela 6 – Resultados das simulações do *testbench* do laboratório de medição de posição angular com contexto direto.

| Iteração | Nota (máx. 8) | Veredito |
|-----------------|----------------------|-----------------|
| 1 | 8 | aceitável |
| 2 | 8 | aceitável |
| 3 | 7 | aceitável |
| 4 | 7 | aceitável |
| 5 | 8 | aceitável |
| 6 | 7 | incorreto |
| 7 | 7 | aceitável |
| 8 | 7 | aceitável |
| 9 | 7 | aceitável |
| 10 | 7 | aceitável |
| 11 | 7 | aceitável |
| 12 | 6 | aceitável |
| 13 | 7 | aceitável |
| 14 | 7 | aceitável |
| 15 | 8 | aceitável |
| 16 | 7 | aceitável |
| 17 | 7 | aceitável |
| 18 | 6 | aceitável |
| 19 | 7 | aceitável |
| 20 | 8 | aceitável |

Fonte: o Autor, 2024.

Tabela 7 – Resultados das simulações do *testbench* do laboratório de medição de posição angular com contexto informativo.

| Iteração | Nota (máx. 8) | Veredito |
|-----------------|----------------------|-----------------|
| 1 | 7 | aceitável |
| 2 | 7 | aceitável |
| 3 | 7 | inviável |
| 4 | 7 | incorreto |
| 5 | 8 | aceitável |
| 6 | 8 | aceitável |
| 7 | 6 | aceitável |
| 8 | 4 | inviável |
| 9 | 6 | aceitável |
| 10 | 7 | aceitável |
| 11 | 7 | aceitável |
| 12 | 4 | inviável |
| 13 | 7 | aceitável |
| 14 | 7 | aceitável |
| 15 | 7 | aceitável |
| 16 | 7 | aceitável |
| 17 | 6 | genérico |
| 18 | 6 | inviável |
| 19 | 6 | aceitável |
| 20 | 7 | aceitável |

Fonte: o Autor, 2024.

Tabela 8 – Dados calculados a partir dos resultados do *testbench* de medição de posição angular

| | Direto | Interativo | Total |
|----------------------|---------------|-------------------|--------------|
| Média | 7,15 | 6,55 | 6,85 |
| Desvio Padrão | 0,5871 | 1,0501 | 0,8930 |
| Ótimo | 0 | 0 | 0 |
| Aceitável | 19 | 14 | 33 |
| Genérico | 0 | 1 | 1 |
| Inviável | 0 | 4 | 4 |
| Incorreto | 1 | 1 | 2 |

Fonte: o Autor, 2024.

Apesar de sutil, percebe-se que o contexto investigativo obteve piores resultados, pois 5 dos 6 vereditos negativos foram direcionados a esse. Além disso, o maior desvio padrão e a menor média indicam a pior consistência de boas notas. Contudo, essa observação poderá apenas ser confirmada após a comparação na seção seguinte.

Algo a destacar nos resultados é a falta de vereditos ditos *ótimos*, apesar do elevado número de notas máximas. Visto que foi solicitado separadamente que o GPT avaliador fornecesse as notas e os vereditos, nem sempre o critério foi o mesmo em ambos, de modo

a não existir uma equivalência. Esse detalhe é um dos primeiros indícios de falhas do modelos, pois, uma vez que os requisitos e o sumário são os mesmos, a interpretação do GPT avaliador deve ser a mesma tanto para a geração das notas quanto dos vereditos.

4.3.2 Laboratório de Termometria

Ao observar as notas e os vereditos expostos nas Tabelas 9 e 10, percebe-se que muitas das notas estão próximas ao máximo de 8. O laboratório de termometria foi o *testbench* mais consistente, com um total de 36 das 40 iterações obtendo a categoria *aceitável*. O remanescente ficou dividido entre vereditos *ótimos* e *inviáveis*, de modo que 38 das iterações foram positivamente categorizadas.

Tabela 9 – Resultados das simulações do *testbench* do laboratório de termometria com contexto direto.

| Iteração | Nota (máx. 8) | Veredito |
|----------|---------------|-----------|
| 1 | 6 | aceitável |
| 2 | 7 | aceitável |
| 3 | 8 | aceitável |
| 4 | 7 | ótimo |
| 5 | 8 | aceitável |
| 6 | 8 | aceitável |
| 7 | 7 | aceitável |
| 8 | 7 | aceitável |
| 9 | 8 | aceitável |
| 10 | 7 | aceitável |
| 11 | 7 | aceitável |
| 12 | 8 | aceitável |
| 13 | 7 | aceitável |
| 14 | 7 | aceitável |
| 15 | 5 | aceitável |
| 16 | 7 | aceitável |
| 17 | 7 | ótimo |
| 18 | 7 | aceitável |
| 19 | 7 | aceitável |
| 20 | 5 | aceitável |

Fonte: o Autor, 2024.

Tabela 10 – Resultados das simulações do *testbench* do laboratório de termometria com contexto informativo.

| Iteração | Nota (máx. 8) | Veredito |
|----------|---------------|-----------|
| 1 | 7 | aceitável |
| 2 | 7 | aceitável |
| 3 | 7 | aceitável |
| 4 | 6 | aceitável |
| 5 | 7 | aceitável |
| 6 | 4 | aceitável |
| 7 | 7 | aceitável |
| 8 | 7 | aceitável |
| 9 | 8 | aceitável |
| 10 | 6 | aceitável |
| 11 | 7 | aceitável |
| 12 | 8 | aceitável |
| 13 | 7 | aceitável |
| 14 | 7 | aceitável |
| 15 | 7 | aceitável |
| 16 | 4 | aceitável |
| 17 | 6 | inviável |
| 18 | 6 | aceitável |
| 19 | 5 | inviável |
| 20 | 6 | aceitável |

Fonte: o Autor, 2024.

Conforme a apresenta a Tabela 11, a média para o contexto direto ficou em 7 e para o investigativo 6,45, em que o desempenho ideal é 8. Observações preliminares do porquê da nota máxima não ter sido tão atingida indicam que o problema da ferramenta, para esse *testbench* foi calcular a corrente máxima através do NTC, impedindo, assim, que o efeito de auto-aquecimento possa ser devidamente considerado.

Tabela 11 – Dados calculados a partir dos resultados do *testbench* de termometria

| | Direto | Interativo | Total |
|----------------------|--------|------------|--------|
| Média | 7 | 6,45 | 6,725 |
| Desvio Padrão | 0,8584 | 1,0990 | 1,0124 |
| Ótimo | 2 | 0 | 2 |
| Aceitável | 18 | 18 | 36 |
| Genérico | 0 | 0 | 0 |
| Inviável | 0 | 2 | 2 |
| Incorreto | 0 | 0 | 0 |

Fonte: o Autor, 2024.

Os resultados obtidos para o laboratório de termometria podem ser considerados um indicativo da capacidade da ferramenta de resolver o problema proposto, porém é

necessário ainda confirmar por meio da validação manual para que se possa analisar se o GPT avaliador julgou corretamente os requisitos. Lembra-se que esse *testbench* possuía o objetivo de testar as características mais qualitativas de uma solução, sem se atrelar muito aos valores numéricos. Este detalhe será importante ao comparar as notas e vereditos entre os projetos testados.

4.3.3 Projeto de Acelerometria

O *testbench* de acelerometria foi o que obteve a menor média, a de 2,925, das quais apenas 3 obtiveram vereditos positivos (ver Tabelas 12, 13 e 14). Esse projeto possui a característica de ser mais voltado a valores quantitativos, com requisitos explicitamente exigindo valores específicos para a capacitância, resistência, ganho, entre outros. O GPT é conhecido por apresentar resultados negativos para problemas matemáticos, porém, ainda assim acreditava-se que o GPT-4-Turbo poderia mais consistentemente acertar nos cálculos.

Tabela 12 – Resultados das simulações do *testbench* do projeto de acelerometria com contexto direto.

| Iteração | Nota (máx. 9) | Veredito |
|----------|---------------|-----------|
| 1 | 4 | inviável |
| 2 | 3 | inviável |
| 3 | 3 | inviável |
| 4 | 3 | genérico |
| 5 | 3 | inviável |
| 6 | 2 | inviável |
| 7 | 3 | inviável |
| 8 | 3 | inviável |
| 9 | 3 | incorreto |
| 10 | 1 | inviável |
| 11 | 4 | genérico |
| 12 | 4 | incorreto |
| 13 | 3 | inviável |
| 14 | 4 | inviável |
| 15 | 2 | incorreto |
| 16 | 7 | aceitável |
| 17 | 3 | inviável |
| 18 | 5 | inviável |
| 19 | 2 | genérico |
| 20 | 3 | inviável |

Fonte: o Autor, 2024.

Tabela 13 – Resultados das simulações do *testbench* do laboratório de acelerometria com contexto informativo.

| Iteração | Nota (máx. 9) | Veredito |
|----------|---------------|-----------|
| 1 | 2 | incorreto |
| 2 | 3 | aceitável |
| 3 | 3 | inviável |
| 4 | 4 | inviável |
| 5 | 4 | inviável |
| 6 | 2 | inviável |
| 7 | 2 | inviável |
| 8 | 3 | aceitável |
| 9 | 2 | inviável |
| 10 | 5 | inviável |
| 11 | 2 | genérico |
| 12 | 3 | incorreto |
| 13 | 2 | genérico |
| 14 | 2 | incorreto |
| 15 | 2 | inviável |
| 16 | 1 | inviável |
| 17 | 2 | inviável |
| 18 | 3 | inviável |
| 19 | 3 | incorreto |
| 20 | 2 | inviável |

Fonte: o Autor, 2024.

Se por um lado o laboratório de termometria apresenta, em sua maioria, vereditos aceitáveis, o *testbench* de acelerometria apresenta o contrário. Com exceção da iteração 17 do contexto direto, que atingiu 7 requisitos e obteve o veredito de aceitável, todas as outras execuções obtiveram uma nota igual ou inferior a 5. Apesar de ser o único que possui tanto a nota, quanto o veredito, positivos, entende-se que isso indica um potencial existente no GPT, que possivelmente será mais proeminente em futuras versões do modelo ou da ferramenta.

Tabela 14 – Dados calculados a partir dos resultados do *testbench* de acelerometria.

| | Direto | Interativo | Total |
|----------------------|--------|------------|--------|
| Média | 3,25 | 2,6 | 2,925 |
| Desvio Padrão | 1,2513 | 0,9403 | 1,1410 |
| Ótimo | 0 | 0 | 0 |
| Aceitável | 1 | 2 | 3 |
| Genérico | 3 | 2 | 5 |
| Inviável | 13 | 12 | 25 |
| Incorreto | 3 | 4 | 7 |

Fonte: o Autor, 2024.

4.3.4 Projeto de Pressão e Temperatura Superficial

O projeto de medição de pressão e temperatura superficial obteve a maior variação de notas e vereditos. As Tabelas 15 e 16 apresentam os dados obtidos, a Tabela 17 expõe algumas estatísticas, em que é possível observar o alto valor de desvio padrão. Diferentemente do *testbench* de termometria, cujo foco era nos aspectos qualitativos, ou de acelerometria, que exigia habilidade matemática, os requisitos do último projeto estavam mais voltados à arquitetura e ao fluxo de projeto. O importante a se considerar nesse *testbench* é se alguns blocos estão presentes, assim como se seus posicionamentos no diagrama de blocos estão adequados. Consequentemente, ocorreram casos em que os objetivos obrigatórios não estavam devidamente cobertos, de modo que 12, das 40 iterações, recebessem uma avaliação negativa. Além disso, o *testbench* de pressão e temperatura superficial era o que contava com o maior número de requisitos, com 12. O de termometria e o de medição de ângulo possuíam 8, enquanto de acelerometria 9. Entende-se, portanto, que o desvio padrão maior é, também, decorrente da maior faixa de valores.

Tabela 15 – Resultados das simulações do *testbench* do projeto de pressão e temperatura superficial com contexto direto.

| Iteração | Nota (máx. 12) | Veredito |
|----------|----------------|-----------|
| 1 | 8 | aceitável |
| 2 | 7 | aceitável |
| 3 | 9 | aceitável |
| 4 | 10 | aceitável |
| 5 | 10 | inviável |
| 6 | 11 | aceitável |
| 7 | 9 | aceitável |
| 8 | 11 | aceitável |
| 9 | 12 | inviável |
| 10 | 11 | aceitável |
| 11 | 10 | incorreto |
| 12 | 11 | aceitável |
| 13 | 11 | aceitável |
| 14 | 10 | incorreto |
| 15 | 8 | aceitável |
| 16 | 10 | incorreto |
| 17 | 11 | aceitável |
| 18 | 11 | aceitável |
| 19 | 6 | aceitável |
| 20 | 7 | incorreto |

Fonte: o Autor, 2024.

Tabela 16 – Resultados das simulações do *testbench* do laboratório de pressão e temperatura superficial com contexto informativo.

| Iteração | Nota (máx. 12) | Veredito |
|-----------------|-----------------------|-----------------|
| 1 | 9 | incorreto |
| 2 | 10 | aceitável |
| 3 | 9 | aceitável |
| 4 | 10 | aceitável |
| 5 | 8 | inviável |
| 6 | 10 | aceitável |
| 7 | 7 | inviável |
| 8 | 8 | aceitável |
| 9 | 11 | aceitável |
| 10 | 7 | incorreto |
| 11 | 10 | aceitável |
| 12 | 8 | aceitável |
| 13 | 7 | aceitável |
| 14 | 7 | incorreto |
| 15 | 9 | aceitável |
| 16 | 10 | aceitável |
| 17 | 11 | aceitável |
| 18 | 10 | aceitável |
| 19 | 8 | aceitável |
| 20 | 7 | incorreto |

Fonte: o Autor, 2024.

Tabela 17 – Dados calculados a partir dos resultados do *testbench* de pressão e temperatura superficial

| | Direto | Interativo | Total |
|----------------------|---------------|-------------------|--------------|
| Média | 9,65 | 8,8 | 9,225 |
| Desvio Padrão | 1,6631 | 1,3992 | 1,5769 |
| Ótimo | 0 | 0 | 0 |
| Aceitável | 14 | 14 | 28 |
| Genérico | 0 | 0 | 0 |
| Inviável | 2 | 2 | 4 |
| Incorreto | 4 | 4 | 8 |

Fonte: o Autor, 2024.

Um caso a se destacar é o da iteração 9 de contexto direto, pois a nota foi máxima, ou seja, todos os requisitos foram cumpridos, mas o veredito atribuído foi o de *inviável*. Em consequência dessas avaliações serem feitas em duas etapas, o modelo não necessariamente julga a solução de forma igual. Nesse caso em específico, a resposta textual deixou implícito que o filtro passa-baixas vinha antes do multiplexador, além de não mencionar o uso de pontes de Wheatstone após os extensômetros. Ao dar a nota, o GPT avaliador, por algum

motivo, não detectou essas falhas e atribuiu um score ótimo. No entanto, ao dar o veredito, o entendimento foi outro.

4.4 Validação manual

Após a leitura da subseção anterior, é possível inferir um panorama positivo em relação aos dados obtidos. Ou seja, a partir dos testes e das avaliações automatizadas, pode-se entender que o modelo possui capacidade de auxiliar no projeto de sistemas de aquisição de sinais, mesmo havendo um recorte somente para alguns cenários. Portanto, pode-se seguir com a validação para entender se essa consideração se mantém perante a análise manual.

Uma vez que expor individualmente os erros encontrados em cada uma das iterações seria extenso e demasiadamente repetitivo, serão apresentados os destaques positivos e negativos encontrados após a análise manual dos resultados. Em casos de *outliers*, será informado o número da iteração para que se possa entender como a nota e o veredito foram afetados.

É pertinente destacar que o trabalho procurou usufruir de todas as técnicas que estavam ao seu alcance, mas existem inevitáveis limitações tecnológicas no processo de construir a ferramenta, uma vez que, o objetivo verdadeiro das LLMs é, como sugere o nome, linguagem. Contudo, entende-se que a exposição dos erros encontrados servem como referência para futuros trabalhos que possam tentar utilizar o que o projeto está propondo.

4.4.1 Laboratório de Medição de Posição Angular

As notas e os vereditos obtidos para o *testbench* de posição angular foram, na sua maioria, positivos, conforme indicam as Tabelas 6, 7 e 8. No entanto, ao analisar de forma mais aprofundada as soluções propostas pela ferramenta, percebe-se que existem detalhes que vão contra essa afirmação.

Visto que o foco dessas execuções não era o de cálculo, a grande parte dos problemas que podem ser observados estão no âmbito das escolhas de projeto que foram feitas pelo modelo. Não teoricamente erradas, às vezes apenas não fazem sentido perante o que foi proposto. Um exemplo recorrente são as escolhas dos modelos dos componentes, pois em 20 soluções foi especificado o uso de amplificadores operacionais *rail-to-rail* para tensões de entrada e de saída abaixo da tensão de alimentação, de modo a não justificar o uso. Um lado positivo é que em todas as iterações foi informado o modelo dos AMPOPs, que favorece o quesito de propor soluções detalhadas, mas nem todas as escolhas parecem justificadas. Em 24 casos foi recomendado o uso do *TL072* e outros da mesma família, que são bastante antigos e podem não ser ideais para aplicações de precisão. Além disso, houveram situações em que estes AMPOPs foram utilizados em combinação com componentes de precisão, ou

seja, parte da arquitetura utilizava modelos antigos de uso-geral, enquanto a outra parte utilizava modelos especializados. Supõe-se que o motivo por trás dessas escolhas é devido às suas características genéricas, de modo que existe mais informação online disponível a respeito dessas, tornando-as mais predominantes na etapa de pré-treinamento do GPT. Esses comentários referentes aos componentes são válidos para todos os *testbenches* e, por conseguinte, não serão novamente discutidos.

Um outro problema proeminente foi o da equação errada para topologia não-inversora de amplificador, em que o ganho é descrito corretamente por

$$G = 1 + \frac{R_f}{R_g}, \quad (16)$$

em que R_f é a resistência de *feedback* e R_g está entre o terminal inversor do AMPOP e o terra.

Em 7 das 40 iterações, sem diferenciar entre contextos, a ferramenta utilizou essa topologia com o intuito de obter ganhos inferiores a 1, o que é matematicamente impossível. Além disso, apesar de explicitamente requisitar o amplificador não-inversor e apresentar a fórmula correta, os valores dos resistores apenas resultariam no ganho desejado caso fosse um inversor. Uma vez que o projeto não explicita a polaridade do sinal, a escolha de um inversor seria aceitável. Esse erro matemático foi apenas identificado pelo GPT avaliador na iteração 8 do cenário direto, em que foi corretamente atribuído o veredito de *incorreto*.

Um destaque positivo das respostas foi a da escolha correta dos valores de resistores e capacitores para os filtros implementados. Em 14 iterações, foi explicitado os componentes para implementar filtros passa-baixas do tipo Butterworth. Em 11 dessas, a frequência de corte é de fato o que se obtém caso os resistores e capacitores sugeridos sejam utilizados. Entretanto, em 3 execuções os valores estavam projetados para uma frequência de corte 10 vezes maior ou menor que a indicada. Testes preliminares também haviam indicado que um erro comum dos modelos GPT ao realizar cálculos é a escala do resultado.

Ainda em relação aos filtros, uma característica clara dos GPTs é sua dificuldade de tomar decisões a partir de informações implícitas. Por exemplo, sabe-se que frequências de 50 e 60 Hz devem ser atenuadas, além disso, deve-se utilizar um filtro *anti-aliasing*, pois, em determinado projeto, a taxa de amostragem é de 1 kHz. Contudo, em vez de implementar um único filtro passa-baixas por volta de 10 Hz, que cumpre todos os requisitos em um bloco, em 3 iterações o modelo propõe 3 blocos de filtragem: um *Notch* para 50 Hz, um *Notch* para 60 Hz e um *anti-aliasing*. Apesar de teoricamente possível, é um sinal de um projeto desnecessariamente complicado. Entende-se que, em determinadas situações, essa redundância é favorável, porém, o *prompt* apenas solicita que frequências em torno de 50 e 60 Hz sejam atenuadas. Como a interpretação é subjetiva, não é possível afirmar que a resposta do GPT está errada, apenas indicativa de como os requisitos são interpretados.

Em 7 situações (todas do contexto investigativo), a implementação proposta para os filtros está incorreta, com a frequência de corte longes do que deveria estar, como, por exemplo, um passa-baixas em 100 Hz cujo objetivo é atenuar ruído em 50 Hz. Um caso que ocorreu 4 vezes com o filtro *anti-aliasing* foi o de seu posicionamento antes de algum outro bloco de condicionamento e não diretamente anterior ao ADC, que não é uma boa prática de projeto. Em 4 instâncias, foram escolhidas frequências de corte próximas ou maiores que 450 Hz, o que está errado, pois o objetivo é atenuar 500 Hz em, no mínimo, 20 dB.

Outro erro é em relação a concordância entre os blocos. Na maioria dos casos, esse problema não é tão evidente, mas em outros é muito, a ponto de remover todo o sentido da solução. Em 3 iterações, o modelo propôs modular o nível de tensão de 5 V para 7 V e, para tal, aplicou um ganho. Contudo, sem o conhecimento que o bloco anterior já havia feito isso, o modelo repetiu o processo, assim amplificando a tensão novamente, de modo a passar dos 7 V. Pode-se argumentar que esse erro é puramente devido ao método na qual a ferramenta foi implementada, pois, como abordado na seção 3.3, o histórico de conversa da etapa de detalhamento não é mantida para outros blocos. Entretanto, lembra-se que o GPT principal passa por uma etapa de revisão ao fim, em que se esperava que erros grosseiros, como esses, pudessem ser remediados, o que não ocorreu. Além disso, a escolha por não manter o histórico completo em todos os momentos deve-se - também - pela otimização de *tokens*, pois, sem ela, o valor de execução seria mais elevado. Assim como em alguns outros erros discutidos, o modelo avaliador não conseguiu inferir essas situações e acreditou nas alucinações descritas pelo GPT principal.

Por fim, serão discutidos os resultados obtidos na diagramação da arquitetura, que são importantes para entender o fluxo escolhido pelo modelo. Devido a característica não determinística dos modelos, a solução é sempre diferente, mas o diagrama de blocos segue um padrão, com algumas iterações escolhendo arquiteturas bastante similares. No geral, os diagramas seguiram um fluxo similar ou igual ao apresentado pela Figura 21³, em que se percebe a escolha dos *Notchs* em conjunto com o passa-baixas.

Figura 21 – Diagrama de blocos usual do laboratório de medição da posição angular obtida da iteração 12 com contexto investigativo.



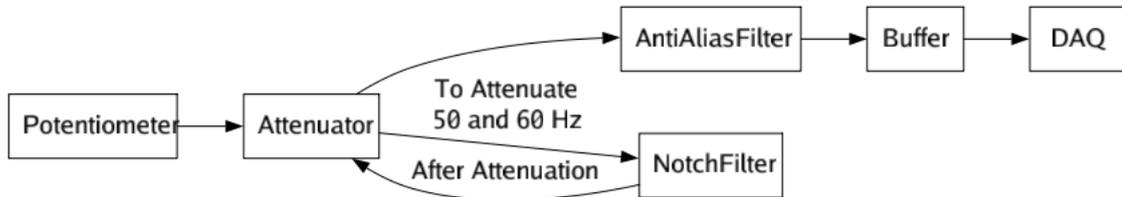
Fonte: o Autor, 2024.

No entanto, algumas arquiteturas não fazem sentido e deixam o usuário confuso em relação ao que está de fato sendo proposto, como é o caso daquela indicada pela Figura 22.

³ Nota-se que o texto não está adequadamente posicionado dentro dos blocos; entretanto, essas imagens foram retiradas diretamente da execução do projeto, para que se possa analisar a apresentação dos resultados.

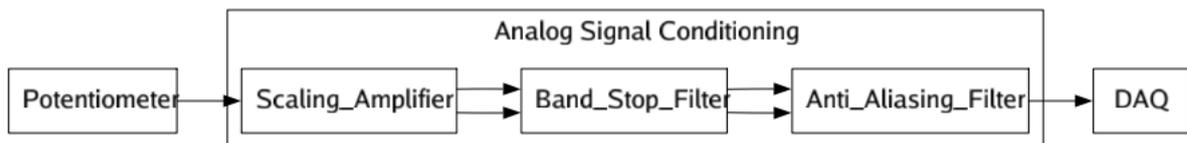
A Figura 23, por outro lado, até apresenta blocos coerentes com o proposto, mas faz uso de flechas duplas, que subentende sinais diferenciais, o que não é mencionado na solução.

Figura 22 – Diagrama de blocos do laboratório de medição da posição angular obtida da iteração 10 com contexto direto, em que o fluxo proposto está incorreto e confuso.



Fonte: o Autor, 2024.

Figura 23 – Diagrama de blocos do laboratório de medição da posição angular obtida da iteração 20 com contexto direto, em que incorretamente está indicada tensão diferencial.



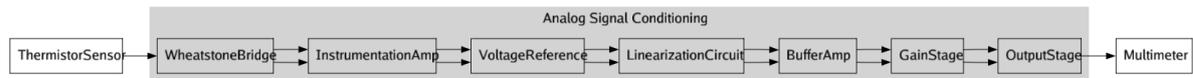
Fonte: o Autor, 2024.

4.4.2 Laboratório de Termometria

O *testbench* de termometria possui a peculiaridade de ser exclusivamente qualitativo (não foram requisitados cálculos de valor de componentes) e, assim, procura validar as habilidades de diagramação e de fluxo lógico da ferramenta. O projeto de medição de posição angular também é mais inclinado a tal, porém restrições com a tensão de saída do sistema exigem com que os ganhos sejam devidamente calculados, assim como as frequências de corte.

Apesar de consistentemente fornecer diagramas de blocos próximos ao esperado, o curioso dos resultados obtidos é a quantidade excessiva de blocos amplificadores. Por exemplo, a Figura 24 apresenta uma arquitetura que corretamente utiliza tanto o resistor de linearização, quanto a ponte de Wheatstone, porém, existem 3 etapas de amplificação: o de instrumentação, um genérico e um *buffer*. Analisando as explicações, percebe-se que todos foram projetados de modo a suficientemente amplificar tudo que é necessário. Dessa forma, o sinal de saída estará cúbico em relação ao que devia. Novamente reforçando o problema de concordância entre os blocos. Nota-se, também, o uso apropriado do indicador de tensão diferencial.

Figura 24 – Diagrama de blocos do laboratório de termometria obtida da iteração 10 com contexto direto, em que incorretamente estão indicadas 3 diferentes etapas de amplificação.



Fonte: o Autor, 2024.

Outro erro nos testes desse *testbench*, foi o uso de amplificadores diferenciais ou de instrumentação sem que o sinal seja de fato diferencial, uma vez que não foi utilizada uma ponte. Apesar de não ser impossível de se fazer, perde-se o sentido de especificar esses amplificadores. Esses casos aconteceram um total 4 vezes, enquanto que o oposto aconteceu 5 vezes, ou seja, foi utilizada uma ponte de Wheatstone mas a topologia de amplificação era não-inversora.

No entanto, de forma geral, pode-se considerar que as notas e os verditos de termometria foram os mais próximos ao ideal, como será visto após apresentar os 2 *testbenches* remanescentes. A Figura 25 foi gerada em uma das iterações de contexto investigativo e está igual, considerando o filtro *anti-aliasing* como opcional, ao diagrama de referência 13, apresentado na seção de metodologia. Em 9 dos 40 casos, a ferramenta corretamente empregou a linearização, a ponte de Wheatstone e a amplificação diferencial. Isso não significa que não houveram outros erros nesses testes, mas de modo geral estão no rumo certo.

Figura 25 – Diagrama de blocos correto do laboratório de termometria obtida da iteração 3 com contexto investigativo.



Fonte: o Autor, 2024.

4.4.3 Projeto de Acelerometria

O diagrama de blocos para o dispositivo medidor de vibrações, dentre os quatro *testbenches*, foi o mais consistente. Visto que os objetivos eram mais voltados aos cálculos, entende-se que as questões de arquitetura foram deixadas mais em aberto, sem que houvessem especificações a respeito da aquisição. No entanto, justamente por isso, uma abordagem utilizando um ADC era esperada, pois é uma solução viável para um dispositivo móvel. Medição direta, por exemplo, não seria adequada dado o caso de uso. As Figuras 26 e 27 apresentam as arquiteturas que, salvo algumas leves variações, foram empregadas em mais da metade das iterações.

Figura 26 – Diagrama de blocos do laboratório de acelerometria obtida da iteração 6 com contexto direto, em que um passa-altas é empregado.



Fonte: o Autor, 2024.

Figura 27 – Diagrama de blocos do laboratório de acelerometria obtida da iteração 10 com contexto investigativo, em que a frequência de 0,25 Hz não é atenuada.

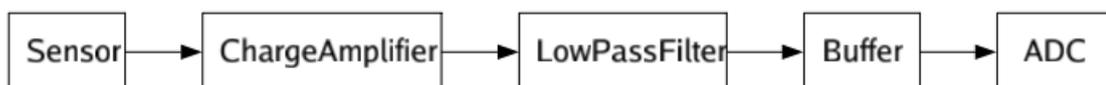


Fonte: o Autor, 2024.

Percebe-se que, na Figura 27, as frequências na faixa de 0,25 Hz não foram atenuadas. Essa omissão foi comum no contexto investigativo - com 10 aparições, no qual o modelo precisaria fazer alguma pergunta relacionada a tal para obter esse requisito de projeto. Além disso, existem outros problemas na escolha dos filtro, pois a faixa de interesse justamente está em torno de 2 Hz, pois não é recomendado projetar um filtro com essa mesma frequência de corte, uma vez que a atenuação será de -3 dB.

Um erro que aconteceu 12 vezes no contexto direto, em que já se possuía o requisito de atenuar 0,25 Hz, foi o uso exclusivo de passa-baixas, como pode ser visto na Figura 28. Nesse diagrama, as frequências de corte são, respectivamente, 0,25 Hz e 2,5 Hz, de modo que a faixa desejada de 2 Hz não estará livre de atenuação. Assim como a discussão referente às escolhas dos componentes no laboratório de posição angular, atribui-se esse erro à maior popularidade de passa-baixas, que diminui a probabilidade do modelo optar por um passa-altas, mesmo que seja a única opção correta nesse caso.

Figura 28 – Diagrama de blocos do laboratório de acelerometria obtida da iteração 4 com contexto investigativo, em que apenas um passa-baixas são empregados.



Fonte: o Autor, 2024.

No entanto, é necessário reforçar novamente a limitação existente da implementação da etapa de detalhamento. Uma vez que os blocos não possuem conhecimento do que ficou decidido para os anteriores, é difícil garantir que existirá consistência na solução. Com as respostas obtidas nesse *testbench*, serão trazidas algumas sugestões para trabalhos

futuros de como isso pode ser corrigido. Contudo, nem todos problemas apresentados são vinculados a isso, de modo que não será suficiente para garantir uma resposta ideal.

Analisando apenas os dados apresentados nas tabelas 12, 13 e 14, o *testbench* de acelerometria apresentou as menores médias e a maior quantidade de vereditos negativos. No entanto, ao observar os resultados com maior profundidade, é possível entender os motivos pelos quais isso ocorreu. De fato as soluções estão incorretas e inviáveis, mas percebe-se que as exigências estão com um escopo bem delimitado, de modo que o modelo avaliador consegue corretamente julgar a solução. Como discutido para as execuções de termometria e de posição angular, as respostas apresentadas nem sempre estavam tão certas quanto as avaliações as faziam parecer, muitas vezes por inconsistências teóricas, que não precisariam ser explicitamente exigidas. Contudo, visto que o o *testbench* de acelerometria forneceu tanto os valores numéricos quanto a margem de erro permitida, o GPT avaliador pôde objetivamente comprovar se o requisito foi cumprido.

Os requisitos exigiam que todo o processo estivesse correto, uma vez todos os componentes precisam estar devidamente dimensionados para que a solução funcione. Assim, observou-se que parte dos cálculos estavam coerentes, mas que detalhes impediam que os requisitos matemáticos fossem atingidos. Um exemplo é em relação à aceleração pico-a-pico, que é necessário para que se possa encontrar a carga e, conseqüentemente, o valor da capacitância e resistência de *feedback* do amplificador. Em 17 casos, a ferramenta considerou um valor padrão de aceleração, pois não percebeu que dada a frequência e a amplitude de oscilação de vibração, poderia-se calcular a aceleração pico a pico esperada. Dessa forma, os cálculos subseqüentes eram todos derivados dessa suposição inicial, que impediam que os requisitos fossem devidamente atingidos.

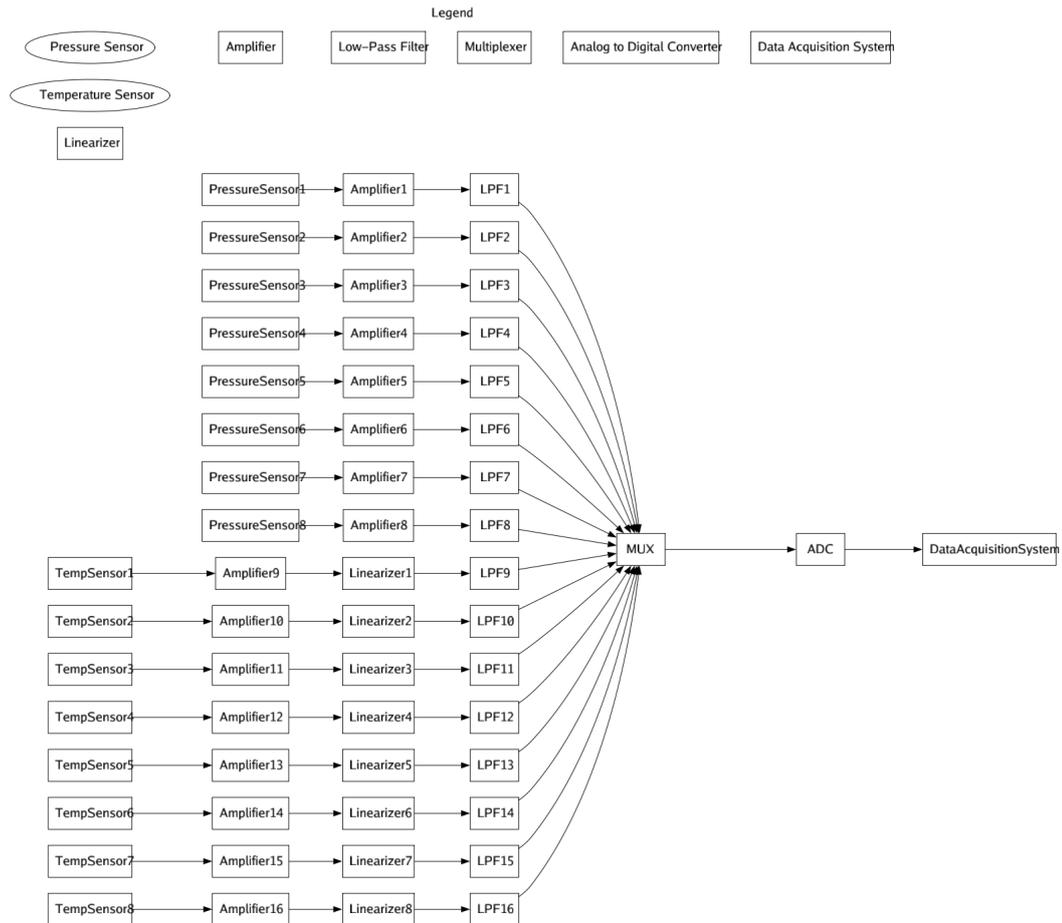
4.4.4 Projeto de Pressão e Temperatura Superficial

Para o último *testbench*, observam-se erros similares àqueles encontrados nos laboratórios de termometria e de posição angular, em que as escolhas tomadas não são justificadas. As execuções desse último projeto possuem o intuito de entender se a ferramenta consegue fornecer arquiteturas coerentes para projetos que possuem um fluxo com mais etapas.

Iniciando por problemas técnicos de implementação da ferramenta, tem-se a questão do projeto ser formado por 16 sensores, que criou certas dificuldades no detalhamento dos blocos. Na teoria, pode-se argumentar que um diagrama completo, como o apresentado na Figura 29, está mais adequado que o apresentado em 30, ainda que o fluxo seja o mesmo. Contudo, visto que a ferramenta realiza o detalhamento para cada um dos blocos da arquitetura individualmente, a arquitetura da Figura 29 exigiria 59 ciclos de perguntas e respostas, mesmo que para blocos iguais entre si. Portanto, foi necessário impedir que o GPT optasse por expandir os arranjos de sensores. Essa limitação foi detectada na fase de

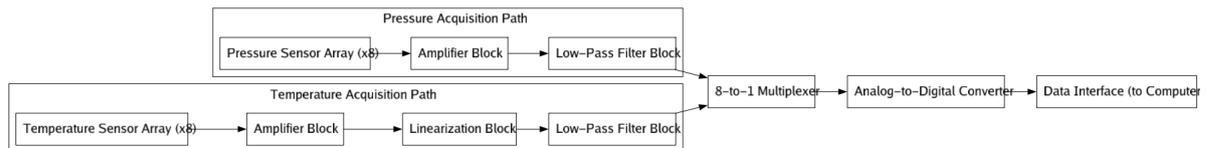
testes preliminares e por isso que existe a frase no *prompt* de arquitetura exigindo que unidades lógicas iguais sejam organizadas em forma de *arrays* (ver apêndice A). Entretanto, nem sempre o modelo sabia como fazer isso e, conseqüentemente, anomalias como a exposta na Figura 31 surgiram em 3 iterações.

Figura 29 – Diagrama de blocos do laboratório de pressão e temperatura superficial retirado de um teste preliminar, em que os arranjos de sensores estão expandidos.



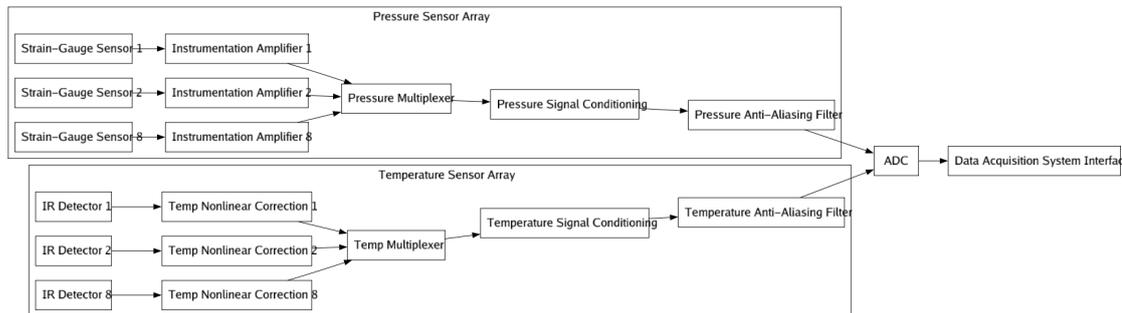
Fonte: o Autor, 2024.

Figura 30 – Diagrama de blocos do laboratório de pressão e temperatura superficial da iteração 7 com contexto investigativo, em que os sensores estão em arranjos.



Fonte: o Autor, 2024.

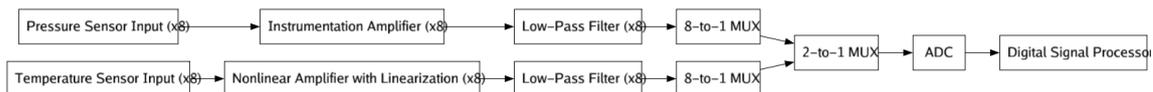
Figura 31 – Diagrama de blocos do laboratório de pressão e temperatura superficial da iteração 16 com contexto direto, em que a apresentação ficou parcialmente expandida.



Fonte: o Autor, 2024.

Em termos de desempenho, percebe-se que em certas execuções o modelo propõe implementações próximas à esperada, mas com erros teóricos. O uso do amplificador de instrumentação para o sensor de temperatura superficial, ou seja, sem que o sinal seja diferencial, ocorreu em 14 iterações. Por outro lado, em 37 execuções (quase todas), o projeto omitiu importantes detalhes da arquitetura, como, por exemplo, o uso de uma ponte de Wheatstone para os extensômetros, que precisaria estar explicitado. Além disso, 12 vezes o multiplexador não considera os 16 canais, como na Figura 30, em que o modelo indica apenas um MUX de 8-1. O correto seria considerar ou um multiplexador de 16-1, ou múltiplos multiplexadores, como na Figura 32.

Figura 32 – Diagrama de blocos do laboratório de pressão e temperatura superficial da iteração 12 com contexto direto, em que os MUXs estão corretamente indicados.



Fonte: o Autor, 2024.

4.4.5 Discussão sobre o Desempenho do Modelo Avaliador

Ao longo das discussões referentes à validação manual, já foram apresentados alguns problemas relacionados ao modelo avaliador. Contudo, para que futuros trabalhos possam atuar diretamente nos erros encontrados, estes serão explicitamente abordados.

O GPT avaliador possui duas atribuições, a de providenciar tanto vereditos quanto notas. O motivo de escolha por separar essas funções foi a dos testes preliminares indicarem inconsistências caso tudo fosse solicitado de uma vez. Entretanto, apesar de ser a melhor alternativa no momento, entende-se que algumas consequências surgiram com essa decisão, como a de diferentes interpretações ao atribuir a nota e o veredito. Exemplos disso são as notas máximas obtidas nos *tetstbenches* que não foram categorizadas como *ótimas*, e vice-versa.

Outro débito técnico em relação ao modelo avaliador é a falta de capacidade de perceber erros teóricos na solução. Em vários momentos, a implementação sugerida continha informações que não faziam sentido, como o uso de amplificadores diferenciais para sinais *single-ended*. Outro exemplo é o uso de filtros cuja frequência de corte estava configurada a estar exatamente em cima da frequência desejada, como no *testbench* de acelerometria.

Assim, um dos principais problema do modelo avaliador é sua permissividade a alucinações contidas na solução, isto é, o modelo acredita nas justificativas fornecidas, mesmo que erradas. O padrão que pode ser percebido nas execuções é que, apesar dos erros, o GPT principal escreve de forma assertiva, como se tudo estivesse correto. Essa característica é comum e inevitável, porém se acreditou que o GPT avaliador conseguiria identificar essas inconsistências. Em alguns casos até foram expostas essas falhas, como na iteração 6 de contexto direto do laboratório de medição de posição angular, pois o erro na equação da topologia não-inversora foi destacado e, conseqüentemente, o veredito de *inviável* foi atribuído.

4.5 Comparação dos Contextos

4.5.1 *Mann-Whitney U-Test*

As limitações observadas no desempenho do modelo avaliador restringem as possíveis interpretações que podem ser realizadas de um teste estatístico sobre seus resultados. Apesar das falhas existentes e dessas capacidades limitadas do GPT, ainda pode-se utilizar o U-Test para comparar os dois contextos ⁴ utilizados.

Embora o número de amostras seja igual à 20, possibilitando que se teste a hipótese a partir do valor de U , o número elevado de empates exige que seja calculado o valor de z utilizando o desvio padrão corrigido para que se possa afirmar com maior robustez estatística se existe diferença entre os contextos.

Portanto, seguiram-se os passos descritos na seção 2.4, considerando a nota para gerar as classificações, para encontrar os valores apresentados nas Tabelas 18-21.

⁴ Lembra-se que no contexto direto, todos os requisitos são fornecidos ao modelo no início, enquanto que no investigativo, o modelo precisa fazer perguntas ao usuário para descobrir quais as restrições de projeto.

Tabela 18 – Valores do U-Test para o *testbench* de medição de posição angular.

| | |
|--------------------------------|---------|
| Rank Sum - Contexto Direto | 476,5 |
| Rank Sum - Contexto Interativo | 343,5 |
| U_{Direto} | 133,5 |
| $U_{Interativo}$ | 266,5 |
| U | 133,5 |
| μ_U | 200 |
| σ_U | 36,9684 |
| $z - value$ | 1,7988 |
| σ_U corrigido | 32,5162 |
| $z - value$ corrigido | 2,0451 |

Fonte: o Autor, 2024.

Tabela 19 – Valores do U-Test para o *testbench* de termometria.

| | |
|--------------------------------|---------|
| Rank Sum - Contexto Direto | 476,5 |
| Rank Sum - Contexto Interativo | 343,5 |
| U_{Direto} | 133,5 |
| $U_{Interativo}$ | 266,5 |
| U | 133,5 |
| μ_U | 200 |
| σ_U | 36,9684 |
| $z - value$ | 1,7988 |
| σ_U corrigido | 32,5162 |
| $z - value$ corrigido | 2,0451 |

Fonte: o Autor, 2024.

Tabela 20 – Valores do U-Test para o *testbench* de acelerometria.

| | |
|--------------------------------|---------|
| Rank Sum - Contexto Direto | 476,5 |
| Rank Sum - Contexto Interativo | 343,5 |
| U_{Direto} | 133,5 |
| $U_{Interativo}$ | 266,5 |
| U | 133,5 |
| μ_U | 200 |
| σ_U | 36,9684 |
| $z - value$ | 1,7988 |
| σ_U corrigido | 32,5162 |
| $z - value$ corrigido | 2,0451 |

Fonte: o Autor, 2024.

Tabela 21 – Valores do U-Test para o *testbench* de pressão e temperatura superficial.

| | |
|--------------------------------|---------|
| Rank Sum - Contexto Direto | 476,5 |
| Rank Sum - Contexto Interativo | 343,5 |
| U_{Direto} | 133,5 |
| $U_{Interativo}$ | 266,5 |
| U | 133,5 |
| μ_U | 200 |
| σ_U | 36,9684 |
| $z - value$ | 1,7988 |
| σ_U corrigido | 32,5162 |
| $z - value$ corrigido | 2,0451 |

Fonte: o Autor, 2024.

Ao considerar um valor de significância de 5%, ou seja, $\alpha = 0,05$, tem-se que o valor de U de referência é 127 (conforme a Tabela 23) e um valor z de referência de 1,96. Definiu-se já que o teste da hipótese será realizado apenas por meio do $z - value$, porém, é válido ainda observar se existe diferença entre métodos. A Tabela 22 indica para quais *testbenches* a hipótese de que ambas amostras não possuem diferença significativa foi rejeitada.

Tabela 22 – Resultados dos testes de significância.

| Testbench | Teste por z |
|-----------------------------------|---------------------|
| Posição Angular | H_0 rejeitada |
| Termometria | H_0 não rejeitada |
| Acelerometria | H_0 rejeitada |
| Pressão e Temperatura Superficial | H_0 não rejeitada |

Fonte: o Autor, 2024.

Percebe-se que, ao corrigir os empates por meio do valor z , as hipóteses *testbenches* de acelerometria e de posição angular foram rejeitadas. Essa conclusão faz sentido especialmente para os testes de acelerometria, pois somente é possível calcular tudo que é necessário caso a maioria dos requisitos sejam informados, de modo que, no contexto investigativo, o modelo somente possuirá a chance de corretamente propor uma solução caso faça todas as perguntas necessárias. Desas maneira, o contexto direto possui vantagem.

O laboratório de termometria, por outro lado, possuía o caráter mais qualitativo, de modo que em ambos nem todas as informações adicionais fossem mandatórias. O fato da hipótese do *testbench* de posição angular ter sido rejeitada foi uma surpresa, mas uma possibilidade é a questão do limite máximo de tensão aceito pelo DAQ ser 7 V. Esse detalhe era mandatório e precisava ser extraído por meio de perguntas para o contexto investigativo.

4.6 Custo e Consumo de *Tokens*

Em decorrência do elevado número de chamadas à API da OpenAI durante uma execução da ferramenta, em que cada uma contém o histórico de conversa, o custo de para o projeto é considerável, especialmente tendo em mente o número de iterações realizadas. Decidiu-se incluir essa informação no trabalho para discutir o valor associado a pesquisas que utilizam LLM no seu estado da arte.

Para as 160 iterações, o total gasto para utilizar o *gpt-4-1106-preview* foi de 132,17 USD e, ao adicionar os custos associados com testes preliminares, tem-se um total de 172,37 USD. Portanto, calcula-se que a média por iteração seja em torno de 0,826 USD, porém sabe-se que esse valor depende do *testbench* em questão, pois alguns possuem mais blocos e, assim, exigem mais requisições à OpenAI.

Outro dado interessante a se destacar é número total de *tokens* utilizados ao longo dos testes. No total (testes preliminares mais testes reais) foram trocados ao todo um número de 12.035.022 de *tokens*, dos quais 10.170.950 foram de entrada - que inclui o histórico. Toda essa quantidade de informação foi feita através de aproximadamente 9.230 requisição HTTP.

5 Conclusões

LLMs são o estado da arte de inteligência artificial, pois apenas há algumas décadas atrás dificilmente se imaginaria o tanto de tarefas que esses modelos poderiam cumprir. Sem contar, também, a simplicidade e casualidade de uso. No entanto, apesar da tecnologia por trás dos GPTs, não é possível que se assera sua capacidade para auxílio a projeto de sistemas de aquisição de sinais.

Embora ao longo dos testes os modelos tivessem gerado alucinações (informações incorretas ditas como verdade), todos os requisitos foram atingidos em pelo menos uma das iterações. Assim, entende-se que o real problema é a consistência e a habilidade de sistematicamente providenciar a solução correta. Mesmo no *testbench* de acelerometria, em que grande parte das notas e dos vereditos foram negativos, houveram situações em que, dados todos os detalhes necessários, a ferramenta foi capaz de calcular corretamente parte dos valores exigidos.

Outras questões, como as implementacionais, também devem ser considerados ao julgar as habilidades das LLMs. A ferramenta propõe a geração de toda a síntese a nível sistema, de modo a gerar tanto a arquitetura quanto a implementação de cada bloco. Para que isso possa acontecer, uma série de interações com o GPT devem ser feitas, tornando a conversa extensa. É sabido que modelos de linguagens sofrem quando o contexto da mensagem é amplo, fato que pode ter servido como agravante para as escolhas de projeto tomadas. Esse é o motivo pelo qual a etapa de revisão foi criada, para que o GPT avaliador pudesse mais objetivamente julgar a resposta. Contudo, sistemas de aquisição de sinais são caracteristicamente detalhados, tornando difícil o desenvolvimento de uma ferramenta capaz de auxiliar no fluxo completo.

A metodologia empregada para o projeto foi pensada para que uma ferramenta de uso geral (no área de sistemas de aquisição de sinal) pudesse ser desenvolvida, ou seja, o objetivo não era testar as habilidades do GPT especificamente para os quatro *testbenches* selecionados, pois não se desejava criar situações de *overfitting*. Os *prompts* e o fluxo de projeto não foram desenvolvidos com o intuito de serem ideais para esses quatro projetos e sim para qualquer projeto, independente se o critério do usuário era mais quantitativo ou mais qualitativo.

Em relação ao desempenho da ferramenta em si, entende-se que existem falhas que podem ser melhoradas, principalmente em relação ao modelo avaliador. No entanto, visto que essas limitações são partes apenas do escopo de testes automatizados, entende-se que a implementação em código cumpriu bem sua função e conseguiu extrair as funcionalidades do GPT, mesmo que este possua uma limitação. No entanto, a estratégia de histórico foi a

causa de problemas de concordância entre blocos em múltiplas execuções, pois caso os detalhamentos anteriores fossem mantidos na conversa de uma forma ou de outra, seria possível que alguns cenários obtivessem resultados mais convincentes. Entende-se que a investigação sobre como mitigar esse erro pode trazer avanços na pesquisa e uma melhoria na ferramenta.

Ao observar os dois tipos de contextos de testes desenvolvidos, o direto e o investigativo, e o resultado dos testes de hipóteses, percebe-se que a técnica de perguntas funcionou satisfatoriamente. No cenário investigativo, os *testbenches* possuem apenas uma breve descrição do projeto, de modo que todas as restrições devem ser encontradas por meio das perguntas. Como metade das hipóteses foram não-rejeitadas, entende-se que as perguntas conseguiram exercer seu papel em 2 dos testes. Isso ocorreu no *testbench* de pressão e temperatura superficial, em que a ferramenta conseguiu desempenhar de forma estatisticamente similar caso fornecidos todos requisitos no início ou caso fornecida apenas uma única frase de enunciado. Essa implementação é importante em aplicações reais, pois não se pode esperar que o usuário saiba todas as informações que devem ser passadas de antemão. O modelo, portanto, reconhece - até certo ponto - quais perguntas devem ser feitas para apropriadamente delimitar o escopo. A hipótese, contudo, não foi sempre aceita, de modo a existir um espaço de melhoria. De forma geral, no entanto, a proposta das perguntas apresentou potencial, assim como o GPT emulador.

Analisando os objetivos apresentados na seção 1.3, considera-se que todos estes foram atingidos, mas que a hipótese de que GPTs possuem a habilidade de auxiliar projetistas deve ser rejeitada. Entre os objetivos específicos, acredita-se que as contribuições trazidas em relação aos *testbenches* e os testes automatizados utilizando GPTs podem ser úteis para diferentes trabalhos, inclusive para aqueles que procuram desenvolver melhorias em cima do que foi proposto aqui. Em relação ao código ser *open-source*, tudo que foi apresentado nesse trabalho, tanto na metodologia quanto nos resultados, pode ser acessado no repositório disponível em Brown (2024).

Por fim, o trabalho apresenta implementações novas na área de sistemas de aquisição de sinais, porém, com isso, deixa ainda mais lacunas a serem investigadas. A partir das 160 iterações realizadas, devem ser analisadas as soluções para entender como os *prompts* podem ser melhorados, uma vez que estes providenciam as instruções base ao modelo e ditam o fluxo de conversa. Além disso, as técnicas de avaliação e emulação por GPT podem ser mais investigadas para que se tornem mecanismos viáveis para testar aplicações que usam LLMs. Trabalhos futuros podem, também, investigar o uso de projetos externos para auxílio da ferramenta, como o LLMLingua (ver seção 2.5), que permite que se condense as informações de uma mensagem em menos *tokens*.

Referências Bibliográficas

- ADAMOPOULOU, E.; MOUSIADES, L. Chatbots: History, technology, and applications. *Machine Learning with Applications*, v. 2, p. 100006, 2020. ISSN 2666-8270. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2666827020300062>>.
- ALPAC. *Language and Machines: Computers in Translation and Linguistics: a Report*. [S.l.]: National Academy of Sciences, National Research Council, 1966.
- BAHDANAU, D.; CHO, K.; BENGIO, Y. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016.
- BALBINOT, A.; BRUSAMARELLO, V. J. *Instrumentação e Fundamentos de Medidas*. 2. ed. [S.l.: s.n.], 2010. v. 1.
- BETKER, J.; GOH, G.; JING, L.; BROOKS, T.; WANG, J.; LI, L.; OUYANG, L.; ZHUANG, J.; LEE, J.; GUO, Y. *et al.* Improving image generation with better captions. *Computer Science*. <https://cdn.openai.com/papers/dall-e-3.pdf>, 2023.
- BHARDWAJ, D. Getting started with test automation pyramid. *lambdatest.com*. [Online]. Available: <https://www.lambdatest.com/learning-hub/test-pyramid>, 2023. Accessed: 2024-01-13.
- BISHOP, J. *FilterProTM low-pass design tool*. 2002.
- BLOCKLOVE, J.; GARG, S.; KARRI, R.; PEARCE, H. Chip-chat: Challenges and opportunities in conversational hardware design. In: *2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD)*. IEEE, 2023. Disponível em: <<http://dx.doi.org/10.1109/MLCAD58807.2023.10299874>>.
- BROWN, J. P. D. *chattronics*. 2024. Disponível em: <<https://github.com/JonathanPDB/chattronics>>.
- BROWN, T. B.; MANN, B.; RYDER, N.; SUBBIAH, M.; KAPLAN, J.; DHARIWAL, P.; NEELAKANTAN, A.; SHYAM, P.; SASTRY, G.; ASKELL, A.; AGARWAL, S.; HERBERT-VOSS, A.; KRUEGER, G.; HENIGHAN, T.; CHILD, R.; RAMESH, A.; ZIEGLER, D. M.; WU, J.; WINTER, C.; HESSE, C.; CHEN, M.; SIGLER, E.; LITWIN, M.; GRAY, S.; CHESSE, B.; CLARK, J.; BERNER, C.; MCCANDLISH, S.; RADFORD, A.; SUTSKEVER, I.; AMODEI, D. *Language Models are Few-Shot Learners*. 2020.
- CENTER FOR SECURITY AND EMERGING TECHNOLOGY. *Emerging Technology Observatory Country Activity Tracker: Artificial Intelligence*. 2022. Disponível em: <<https://cat.eto.tech/>>. Acesso em: 2024-01-14.
- CHOMSKY, N. *Syntactic Structures*. The Hague: Mouton and Co., 1957.
- CHUNG, J.; GULCEHRE, C.; CHO, K.; BENGIO, Y. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014.
- COHN, M. *Succeeding with agile: software development using Scrum*. [S.l.]: Pearson Education, 2010.

- COKER, M. *Smashwords*. 2009. Disponível em: <<https://www.smashwords.com/>>. Acesso em: 27-01-2024.
- COMMON CRAWL. *Common Crawl*. 2007. Disponível em: <<https://commoncrawl.org>>.
- ELLSON, J.; GANSNER, E. R.; KOUTSOFIOS, E.; NORTH, S. C.; WOODHULL, G. Graphviz and dynagraph — static and dynamic graph drawing tools. In: _____. *Mathematics and Visualization*. Springer Berlin Heidelberg, 2004. p. 127–148. ISBN 9783642186387. Disponível em: <http://dx.doi.org/10.1007/978-3-642-18638-7_6>.
- ELOUNDOU, T.; MANNING, S.; MISHKIN, P.; ROCK, D. *GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models*. 2023.
- FELDMAN, S. *GNU Make*. 1976. Disponível em: <https://www.gnu.org/software/make/manual/html_node/index.html>. Acesso em: 2024-01-11.
- FOOTE, K. D. *A Brief History of Natural Language Processing*. 2023. Disponível em: <<https://www.dataversity.net/a-brief-history-of-natural-language-processing-nlp/>>. Acesso em: 27-01-2024.
- GAO, L.; BIDERMAN, S.; BLACK, S.; GOLDING, L.; HOPPE, T.; FOSTER, C.; PHANG, J.; HE, H.; THITE, A.; NABESHIMA, N.; PRESSER, S.; LEAHY, C. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.
- GERSTLAUER, A.; HAUBELT, C.; PIMENTEL, A. D.; STEFANOV, T. P.; GAJSKI, D. D.; TEICH, J. Electronic system-level synthesis methodologies. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 28, n. 10, p. 1517–1530, 2009.
- GIELEN, G.; MCCONAGHY, T.; EECKELAERT, T. Performance space modeling for hierarchical synthesis of analog integrated circuits. In: *Proceedings. 42nd Design Automation Conference, 2005*. [S.l.: s.n.], 2005. p. 881–886.
- GIRAY, L. Prompt engineering with chatgpt: A guide for academic writers. *Annals of Biomedical Engineering*, Springer Science and Business Media LLC, v. 51, n. 12, p. 2629–2633, jun. 2023. ISSN 1573-9686. Disponível em: <<http://dx.doi.org/10.1007/s10439-023-03272-4>>.
- GRIESEMER, R.; PIKE, R.; THOMPSON, K. *The Go Programming Language*. 2009. Disponível em: <<https://go.dev/>>. Acesso em: 2024-01-11.
- HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Comput.*, MIT Press, Cambridge, MA, USA, v. 9, n. 8, p. 1735–1780, nov 1997. ISSN 0899-7667.
- HU, K. *ChatGPT sets record for fastest-growing user base - analyst note*. 2023.
- HUTCHINS, J. Looking back to 1952: the first mt conference. In: *Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*. [s.n.], 1997. Disponível em: <<https://api.semanticscholar.org/CorpusID:17263523>>.
- JAEGER, H. *Tutorial on Training Recurrent Neural Networks, Covering BPPT, RTRL, EKF and the "echo State Network" Approach*. [S.l.: s.n.], 2005.

- JIANG, H.; WU, Q.; LIN, C.-Y.; YANG, Y.; QIU, L. LlmLingua: Compressing prompts for accelerated inference of large language models. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2023. p. 13358–13376. Disponível em: <<https://aclanthology.org/2023.emnlp-main.825>>.
- JONES, K. S. Natural language processing: A historical review. In: _____. *Current Issues in Computational Linguistics: In Honour of Don Walker*. Springer Netherlands, 1994. p. 3–16. ISBN 9780585359588. Disponível em: <http://dx.doi.org/10.1007/978-0-585-35958-8_1>.
- KIM, Y.; DENTON, C.; HOANG, L.; RUSH, A. M. *Structured Attention Networks*. 2017.
- KUNDERT, K. Life after spice. *IEEE Solid-State Circuits Magazine*, v. 3, n. 2, p. 23–26, 2011.
- LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, 1998.
- LEHMANN, E. L. *Nonparametrics: Statistical Methods Based on Ranks*. 1. ed. [S.l.]: Springer New York, NY, 1975.
- LO, A. W.; SINGH, M. *et al.* From eliza to chatgpt: The evolution of nlp and financial applications. 2023.
- LOCKE, W.; BOOTH, A. Mechanical translation. 1955.
- LONGLLMLINGUA: Accelerating and Enhancing LLMs in Long Context Scenarios via Prompt Compression", author = "Huiqiang Jiang and Qianhui Wu and Xufang Luo and Dongsheng Li and Chin-Yew Lin and Yuqing Yang and Lili Qiu. *ArXiv preprint*, abs/2310.06839, 2023. Disponível em: <<https://arxiv.org/abs/2310.06839>>.
- MANN, H. B.; WHITNEY, D. R. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, Institute of Mathematical Statistics, v. 18, n. 1, p. 50 – 60, 1947. Disponível em: <<https://doi.org/10.1214/aoms/1177730491>>.
- MCCARTHY, J.; MINSKY, M. L.; ROCHESTER, N.; SHANNON, C. E. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI Magazine*, v. 27, n. 4, 2006. Disponível em: <<https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1904>>.
- MEKOUAR, D. *Here Are Wikipedia's Most Searched Topics in 2023*. 2023. Disponível em: <<https://www.voanews.com/a/here-are-wikipedia-s-most-searched-topics-in-2023-/7385418.html>>. Acesso em: 2024-01-14.
- OPENAI. *Tokenizer*. 2021. Disponível em: <<https://platform.openai.com/tokenizer>>. Acesso em: 2024-01-14.
- OPENAI. *Introducing ChatGPT*. 2022. Disponível em: <<https://openai.com/blog/chatgpt>>. Acesso em: 2024-01-14.
- OPENAI. *GPT-4 Technical Report*. arXiv, 2023. Disponível em: <<https://arxiv.org/abs/2303.08774>>.

OPENAI. *Models*. 2023. Disponível em: <<https://platform.openai.com/docs/models>>. Acesso em: 2024-01-14.

OPENAI. *New models and developer products announced at DevDay*. 2023. Disponível em: <<https://openai.com/blog/new-models-and-developer-products-announced-at-devday>>. Acesso em: 2024-01-21.

OPENAI. *Pricing*. 2023. Disponível em: <<https://openai.com/pricing>>. Acesso em: 2024-01-12.

OPENAI. *TTS*. 2023. Disponível em: <<https://platform.openai.com/docs/models/tts>>. Acesso em: 2024-01-14.

OSIKA, A. *gpt-engineer*. 2023. Disponível em: <<https://github.com/AntonOsika/gpt-engineer>>.

PALLAS-ARENY, R.; WEBSTER, J. G. *Sensors and signal conditioning*. [S.l.]: John Wiley & Sons, 2012.

PHAM, C. M.; HOYLE, A.; SUN, S.; IYYER, M. *TopicGPT: A Prompt-based Topic Modeling Framework*. 2023.

POIBEAU, T. The 1966 alpaca report and its consequences. In: _____. *Machine Translation*. [S.l.: s.n.], 2017. p. 75–89.

PORIA, S.; CAMBRIA, E.; GELBUKH, A. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, v. 108, p. 42–49, 2016. ISSN 0950-7051. New Avenues in Knowledge Bases for Natural Language Processing. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0950705116301721>>.

PYTHAGORA-IO. *gpt-pilot*. 2023. Disponível em: <<https://github.com/Pythagora-io/gpt-pilot>>.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. *et al.* Improving language understanding by generative pre-training. OpenAI, 2018.

RADFORD, A.; WU, J.; CHILD, R.; LUAN, D.; AMODEI, D.; SUTSKEVER, I. *et al.* Language models are unsupervised multitask learners. *OpenAI blog*, v. 1, n. 8, p. 9, 2019.

REDDIT INC. *Reddit*. 2005. Disponível em: <www.reddit.com>. Acesso em: 27-01-2024.

ROSSUM, G. V.; DRAKE, F. L. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

SANI, F.; TODMAN, J. *Experimental Design and Statistics for Psychology, Appendix 1: Statistical Tables*. John Wiley Sons, Ltd, 2006. 183-196 p. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470776124.app1>>.

SCHWAB, K. *The fourth industrial revolution*. [S.l.]: Currency, 2017.

SUTHERLAND, J.; SUTHERLAND, J. *Scrum: the art of doing twice the work in half the time*. [S.l.]: Currency, 2014.

TAFFERNER, Z.; BALÁZS, I.; KRAMMER, O.; GÉCZY, A. Can chatgpt help in electronics research and development? a case study with applied sensors. *Sensors*, v. 23, n. 10, 2023. ISSN 1424-8220. Disponível em: <<https://www.mdpi.com/1424-8220/23/10/4879>>.

TAYLOR, H. R. *Data Acquisition for Sensor Systems*. GBR: Chapman & Hall, Ltd., 1997. ISBN 0412785609.

TURING, A. M. Computing machinery and intelligence. *Mind*, Oxford University Press on behalf of the Mind Association, v. 59, n. 236, p. 433–460, 1950. ISSN 00264423. Disponível em: <<http://www.jstor.org/stable/2251299>>.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. arXiv, 2017. Disponível em: <<https://arxiv.org/abs/1706.03762>>.

WEBER, T. O. *Síntese de CIs analógicos em nível de circuito e sistema utilizando métodos modernos de otimização*. Tese (Doutorado) — Universidade de Sao Paulo, Agencia USP de Gestao da Informacao Academica (AGUIA), 2015. Disponível em: <<http://dx.doi.org/10.11606/T.3.2016.tde-22072016-151343>>.

WEIZENBAUM, J. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 9, n. 1, p. 36–45, jan 1966. ISSN 0001-0782. Disponível em: <<https://doi.org/10.1145/365153.365168>>.

WHITE, J.; FU, Q.; HAYS, S.; SANDBORN, M.; OLEA, C.; GILBERT, H.; ELNASHAR, A.; SPENCER-SMITH, J.; SCHMIDT, D. C. *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT*. 2023.

WHITE, J.; HAYS, S.; FU, Q.; SPENCER-SMITH, J.; SCHMIDT, D. C. *ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design*. 2023.

Wikipedia contributors. *Plagiarism* — *Wikipedia, The Free Encyclopedia*. 2004. Disponível em: <<https://en.wikipedia.org/w/index.php?title=Plagiarism&oldid=5139350>>. Acesso em: 2024-01-15.

ZHU, L.; WANG, X.; WANG, X. Judgelm: Fine-tuned large language models are scalable judges. 2023.

APÊNDICE A – *Prompts*

A.1 Arquitetura

Listing A.1 – Prompt de sistema para elaboração de perguntas da etapa de desenvolvimento da arquitetura.

```
You are an expert analog instrumentation electronics engineer.
You will be proposed a project idea and you will ask essential technical
  questions to further understand the problem.
All conditioning is done in the analog domain.
Your first question will always provide you information on the type of sensors
  the user will use and in this first question you will always provide 2
  suggestions.
With the questions you must know if there is any digital part to the project.
Do not ask irrelevant questions.
Do not ask more than 5 questions.
```

Provide the questions inside a JSON object in the EXACT following format where all caps words are placeholders, that you must fill in:

```
{
  "questions": [
    "1. QUESTION-1",
    "2. QUESTION-2",
    "3. QUESTION-3",
    "4. QUESTION-4",
    "5. QUESTION-5",
  ]
}
```

Listing A.2 – Prompt de sistema para elaboração da arquitetura da solução, incluindo o diagrama de blocos.

```
You are an expert analog instrumentation electronics engineer.
You will be presented with a project idea and you will propose a high level
  architecture of the project.
All conditioning is done in the analog domain.
```

The diagram must be of the effective signal path, therefore, do not include block such as "power supply", "biasing" or any other blocks that dont describe the signal path, unless they have some sort of important detail.

If the project contains an array of the same block, consider them a single block in the design

If the project has an ADC, strongly consider the use of an anti-aliasing filter
You will provide a dot file with the block diagram followed by an explanation of each block.

The explanations you provide should contain the objectives of the block, as well as main functionalities.

You will follow the exact following template where both the graphing and explanations should be in their own markdown block.

The all caps words represent placeholder:

```
```graph
DOT FILE HERE
```
```

at least 2 newlines between the markdown blocks

```
```Explanations
EXPLANATIONS HERE
```
```

Listing A.3 – Prompt a ser concatenado com o feedback para compor a mensagem de usuário na etapa de desenvolvimento da arquitetura.

Your diagram is not good enough.

Review your answers to make sure that all topics and requirements are being taken into account.

Remember to respond in the EXACT template that was previously given.

Please provide a better suggestion, taking into account the following comments regarding what you responded:

A.2 Categorização

Listing A.4 – Prompt de usuário para categorização dos blocos da arquitetura.

With the diagram block proposed, classify each block in one of the following categories:

- A) Sensor
- B) Signal conditioning
- C) Amplification
- D) Filtering
- E) Other conditioning
- F) Direct measurement
- G) ADC
- H) Digital processing
- I) Other

Provide the relationship in the exact following format, where the BLOCK placeholder should be replaced with the name of the block and the CATEGORY placeholder should be replaced with the letter of the category of that block.

```
```Categories
BLOCK1=CATEGORY1
BLOCK2=CATEGORY2
BLOCK3=CATEGORY3
ETC
```
```

Example:

```
```Categories
Sensor=A
Low Pass Filter=D
Multimeter=F
```
```

A.3 Detalhamento

Listing A.5 – Prompt de sistema para elaboração de perguntas da etapa de detalhamento.

You are an expert analog instrumentation electronics engineer.

You will be requested to ask up to 5 questions regarding one of the proposed blocks.

You will ask questions that aim to improve your capability of providing more details on this part of the architecture.

You must find out what are defining requirements of the block, and if the user does not know, provide them with reasonable values.

Ask questions to guarantee that your block can support the project's limits of temperature, voltage, current, etc.

If the block relies on a specific component (such as the sensor block), you must find out exactly what type and model of the component will be used and provide at least one suggestion.

Do not ask irrelevant questions.

Do not ask more than 5 questions.

Provide the questions inside a JSON object in the EXACT following format where all caps words are placeholders, that you must fill in:

```
{
  "questions": [
    "1. QUESTION-1",
    "2. QUESTION-2",
    "3. QUESTION-3",
    "4. QUESTION-4",
    "5. QUESTION-5"
  ]
}
```

Listing A.6 – Prompt a ser concatenado com o feedback para compor a mensagem de usuário na etapa de detalhamento.

Your solution is not good enough.

Please provide a better suggestion, taking into account the follow comments regarding what you responded.

Review your answers to make sure that all topics and requirements are being taken into account.

Remember to respond in the EXACT template that was previously given.

Listing A.7 – Prompt de usuário para a elaboração de perguntas do bloco de sensor.

You must find out what are defining requirements of the sensor, and if the user does not know, provide them with reasonable values.

You must find out exactly what type and model of sensor will be used and provide at least one suggestion.

You must make sure your sensor can support the project's limits of temperature, voltage, current, etc.

You must make sure that the sensor characteristics are known, such as resolution, sensitivity, etc (if applicable), so that the other parts of the project can be properly defined.

Listing A.8 – Prompt de sistema para elaboração do detalhamento do bloco de sensor.

You are an expert analog instrumentation electronics engineer.
You will take all provided information and suggest the best way to implement the sensor block for which the user just answered questions to.
You will provide information regarding the type of the sensor, the project requirements and what the user has to pay attention to when implementing.
You must provide information regarding the range, accuracy and other characteristics such as resolution and sensitivity.
You may take well based decisions if the required information is not explicit, however, do may not provide inconsistencies that break the project.
You will provide suggestions on topologies as well as values.
You will not ask questions.
You will not provide boilerplate and placeholder text.
You will only talk about the Sensor block.

Your whole answer must be inside a markdown code block, exactly in the following template (all caps words are placeholders):

```
```Details
NAME_OF_THE_BLOCK

SOLUTION_HERE
```
```

Listing A.9 – Prompt de usuário para a elaboração de perguntas do bloco de condicionamento de sinal.

You must find out what are defining requirements of the signal conditioning, and if the user does not know, provide them with reasonable values.
The signal conditioning stage must support the chosen sensor as well as provide the expected output for the next stage.
With these questions you must be able to provide topology suggestions as well as their parameter values.

Listing A.10 – Prompt de sistema para elaboração do detalhamento do bloco de condicionamento de sinal.

You are an expert analog instrumentation electronics engineer.

You will take all provided information and suggest the best way to implement the signal conditioning block for which the user just answered questions to.

You will provide information regarding how this stage alters the signal and how it should be implemented in order to function.

You must provide suggestions on topologies.

You will provide values for the topology's required parameters.

You will only talk about the Signal Conditioning block.

You will not ask questions.

You will not provide boilerplate and placeholder text.

Your whole answer must be inside a markdown code block, exactly in the following template (all caps words are placeholders):

```
```Details
NAME_OF_THE_BLOCK

SOLUTION_HERE
```
```

Listing A.11 – Prompt de usuário para a elaboração de perguntas do bloco de amplificação.

You must find out what are defining requirements of the amplification stage, and if the user does not know, provide them with reasonable values.

The amplification stage is critical to define the voltage ranges, as well as the projects overall characteristics.

With the answers to the questions you'll make, you must be able to provide topology suggestions as well as provide numeric gain values, and any other vital values for the amplification stage.

You must provide suggestions for the user to choose from, if there are more than 1 viable possibilities.

Listing A.12 – Prompt de sistema para elaboração do detalhamento do bloco de amplificação.

You are an expert analog instrumentation electronics engineer.

You will take all provided information and suggest the best way to implement the Amplification block for which the user just answered questions to.

You will provide information regarding the type of amplification, such as topology and materials.

You must provide numeric values for the amplification gain, based on the information you have.

If this information was not explicitly provided by the user, you must be capable of calculating and taking decisions based on what you know.
You will not ask questions.
You will not provide boilerplate and placeholder text.
You will only talk about the Amplification block.

Your whole answer must be inside a markdown code block, exactly in the following template (all caps words are placeholders):

```
```Details
NAME_OF_THE_BLOCK

SOLUTION_HERE
```
```

Listing A.13 – Prompt de usuário para a elaboração de perguntas do bloco de filtragem.

You will be asked to provide details on the filtering part of the project. You must find out what are defining requirements of the filter, and if the user does not know, provide them with reasonable values.
With the answers to the questions you'll make, you must be able to provide topology suggestions for the filter, as well as provide numeric values of order, cutoff frequencies, overall characteristics and damping values for important frequency values. Basically, with these questions you must be able to provide real applicable suggestions for the filter.
You must provide suggestions for the user to choose from, if there are more than 1 viable possibilities.

Listing A.14 – Prompt de sistema para elaboração do detalhamento do bloco de filtragem.

You are an expert analog instrumentation electronics engineer. You will take all provided information and suggest the best way to implement the Filtering block for which the user just answered questions to. You must provide the topology that will be used in the project, as well as information on materials and characteristics.
You must provide numeric values for the type order and cutoff frequencies. You will provide information on the ranges that should be filtered out or kept in.
If this information was not explicitly provided by the user, you must be capable of calculating and taking decisions based on what you know.
You will not ask questions.

You will not provide boilerplate and placeholder text.

You will only talk about the Filtering block.

Your whole answer must be inside a markdown code block, exactly in the following template (all caps words are placeholders):

```
```Details
NAME_OF_THE_BLOCK

SOLUTION_HERE
```
```

Listing A.15 – Prompt de usuário para a elaboração de perguntas do bloco de outros tipos de condicionamento.

You will be asked to provide details on one of the conditioning parts of the project.

You must find out what are defining requirements of this conditioning block based on previous messages.

With the answers to the questions you'll make, you must be able to provide topology suggestions for this block, as well as get vital numeric values for it.

You must provide suggestions for the user to choose from, if there are more than 1 viable possibilities.

Listing A.16 – Prompt de sistema para elaboração do detalhamento do bloco de outros tipos de condicionamento.

You are an expert analog instrumentation electronics engineer.

You will take all provided information and suggest the best way to implement this conditioning block for which the user just answered questions to.

You will provide information regarding what sort of conditioning this stage refers to.

You must provide a topology to be used, in which there are specifications of materials and values.

If this information was not explicitly provided by the user, you must be capable of calculating and taking decisions based on what you know.

You must provide reasonable numeric values.

You will not ask questions.

You will not provide boilerplate and placeholder text.

You will only talk about this conditioning block in question.

Your whole answer must be inside a markdown code block, exactly in the following template (all caps words are placeholders):

```
```Details
NAME_OF_THE_BLOCK

SOLUTION_HERE
```
```

Listing A.17 – Prompt de usuário para a elaboração de perguntas do bloco de medição direta.

You will be asked to provide details on the measurement part of the project. With the answers to the questions you'll make, you must be able to provide suggestions on the direct measurement stage. You must be provide suggestions on the approach when taking the measurements and also with regard to which equipment to use.

Listing A.18 – Prompt de sistema para elaboração do detalhamento do bloco de medição direta.

You are an expert analog instrumentation electronics engineer. You will take all provided information and suggest the best way to implement the direct measurement block for which the user just answered questions to. You must provide information regarding what sort of measurement should be done, and how it should be done (frequency, cares, etc). You must provide instructions on how to perform these measurements. You must provide suggestions on the equipment to use. You will not ask questions. You will not provide boilerplate and placeholder text. You will only talk about the direct measurement block.

Your whole answer must be inside a markdown code block, exactly in the following template (all caps words are placeholders):

```
```Details
NAME_OF_THE_BLOCK

SOLUTION_HERE
```
```

Listing A.19 – Prompt de usuário para a elaboração de perguntas do bloco de ADC.

You will be asked to provide details on the ADC.

This is a vital part of the project, as it converts the information into another domain, therefore, with the answers to the questions you'll make, you must be able to provide topology suggestions to the user, and also provide numeric values for the vital parameters of the ADC. Therefore, you must find out what the best approach for the ADC is, taking into account the project requirements.

Listing A.20 – Prompt de sistema para elaboração do detalhamento do bloco de ADC.

You are an expert analog instrumentation electronics engineer.
You will take all provided information and suggest the best way to implement the ADC block for which the user just answered questions to.
You must provide information regarding the type of ADC to use, such as topology.
You must provide numeric information on the required values of sampling rate, resolution, accuracy, speed and other valuable information.
If this information was not explicitly provided by the user, you must be capable of calculating and taking decisions based on what you know.
You will not ask questions.
You will not provide boilerplate and placeholder text.
You will only talk about the ADC block.

Your whole answer must be inside a markdown code block, exactly in the following template (all caps words are placeholders):

```
```Details
NAME_OF_THE_BLOCK

SOLUTION_HERE
```
```

Listing A.21 – Prompt de usuário para a elaboração de perguntas do bloco de processamento digital.

You will be asked to provide details on the digital processing part of the project.
With the answers to the questions you'll make, you must be able to provide information and suggestions on how to approach this stage.
As your focus is not the digital analysis, you must find out more information on how this influences the implementation aspects of the project.

Listing A.22 – Prompt de sistema para elaboração do detalhamento do bloco de processamento digital.

You are an expert analog instrumentation electronics engineer.
 You will take all provided information and suggest the best way to implement the Digital Processing block for which the user just answered questions to.
 You will provide information regarding essential details that have to be covered digitally.
 You will provide suggestions of software and apps that can be used for this.
 You will not provide actual code or any sort of real implementation to this part.
 If this parts contains any sort of filtering, amplification or similar things, you must suggest frequencies to cutoff, amplification gains, etc.
 You will not ask questions.
 You will not provide boilerplate and placeholder text.
 You will only talk about the Digital Processing block.

Your whole answer must be inside a markdown code block, exactly in the following template (all caps words are placeholders):

```
```Details
NAME_OF_THE_BLOCK

SOLUTION_HERE
```
```

Listing A.23 – Prompt de usuário para a elaboração de perguntas do bloco de blocos que não se enquadraram em nenhum.

You will be asked to provide details on the on the blocks of the project.
 With the answers to the questions you'll make and with the information you've gathered from previous messages,
 you must find out what are defining requirements of this block and you must be able to provide topology and implementation suggestions

Listing A.24 – Prompt de sistema para elaboração do detalhamento do bloco de outros tipos de blocos que não se enquadraram em nenhum.

You are an expert analog instrumentation electronics engineer.
 You will take all provided information and suggest the best way to implement this other block for which the user just answered questions to.
 This block did not fit into any specific category, but make to provide the essentials details needed to cover this stage.
 Provide topology suggestions and numeric values if applicable.

If this information was not explicitly provided by the user, you must be capable of calculating and taking decisions based on what you know.
You will not ask questions.
You will not provide boilerplate and placeholder text.
You will provide suggestions on topologies.
You will only talk about this block.

Your whole answer must be inside a markdown code block, exactly in the following template (all caps words are placeholders):

```
```Details
NAME_OF_THE_BLOCK
```
```

A.4 Revisão

Listing A.25 – Prompt de usuário para que correções da solução desenvolvida possam ser feitas.

You are an expert analog instrumentation electronics engineer.
You were presented with a project idea and proposed a high level architecture of the project as well as implementation details.
You will now have a final change to review all your answers in sight of what the user requested.
You must provide correction, if any, to any part of your solution.
All corrections you provide must explain what you want to change, why and how.

Your whole answer must be inside a markdown code block, exactly in the following template

```
```Review
YOUR_CORRECTIONS_GO_HERE
```
```

Listing A.26 – Prompt de usuário para a elaboração do resumo da solução.

Now provide a compilation of the entire solution provided by you throughout the whole conversation.
Don't repeat the information, but also do not remove important details that were provided.
Make sure that everything is in the correct order to be presented to the user.

Include component values, component name/model, required techniques to use, etc.

Include entire calculations, showing the steps that were taken and so on.

Include ANYTHING that is related to numeric values, even if its side information.

Do not include user messages, boilerplate info, etc

Start with a title.

Your answer must be organized based on the architecture blocks.

Your answer must be in exactly the following template, where all caps words are placeholders that you must fill in:

```Summary

TITLE HERE

SUMMARY HERE

```

APÊNDICE B – Melhores Resultados

B.1 Laboratório de Medição de Posição Angular

Listing B.1 – Melhor resultado para o *testbench* de posição angular, obtida na iteração 18 de contexto direto.

Pendulum Angle Measurement System Design

This summary compiles all the details and specifications for the design and implementation of a pendulum angle measurement system using a potentiometer, signal conditioning, filtering, and a data acquisition (DAQ) system.

****Potentiometer (Sensor)****

- Type: Linear rotary potentiometer.
- Value: 10 kOhms.
- Model Suggestion: Bourns 3590 Precision Potentiometer.
- Mounting: Securely attached to the wooden structure with the shaft coupled to the pendulum beam.
- Voltage Range: Expected to vary linearly with the angle of the pendulum, estimated between -5V and +5V for 45 to 135.

****Buffer (Signal Conditioning)****

- Topology: Voltage Follower Configuration using an Operational Amplifier (Op-Amp).
- Op-Amp Suggestion: Texas Instruments OPA340 or Analog Devices AD8605.
- Power Supply: 10V (matching the potentiometer's power supply).
- Input and Output Resistances: High input impedance (>1 M) and low output impedance (<100) to prevent loading and ensure signal integrity.

****Gain Stage (Signal Conditioning)****

- Topology: Non-Inverting Op-Amp Configuration.
- Required Gain (A): $\left(A = \frac{V_{\text{out,max}} - V_{\text{out,min}}}{V_{\text{in,max}} - V_{\text{in,min}}} = \frac{7V - (-7V)}{5V - (-5V)} = 1.4 \right)$.
- Resistor Values: R1 = 4k, R2 = 10k (precision resistors with a 1 percent tolerance).
- Power Supply: 10V (dual supply).

****Band-Pass Filter (Signal Conditioning)****

- Active Twin-T Notch Filter Topology for 50 Hz and 60 Hz attenuation.
- Notch Filter Component Values: $R = 31.8 \text{ k}$, $C = 100 \text{ nF}$ (for 50 Hz), $R = 26.5 \text{ k}$, $C = 100 \text{ nF}$ (for 60 Hz).
- Followed by a Sallen-Key Low-Pass Filter with a Butterworth response.
- Low-Pass Filter Component Values: $R = 3.9 \text{ k}$, $C = 1 \text{ nF}$ (for a cutoff frequency of 200 Hz).

****Anti-Aliasing Filter (Signal Conditioning)****

- Sallen-Key Low-Pass Filter Topology.
- Filter Order: Second-order (two-pole) with -12 dB/octave roll-off.
- Cutoff Frequency: Approximately 210 Hz (below half the DAQ's sampling rate).
- Damping Factor: $= 0.707$ (Butterworth response for flat passband).

****Data Acquisition System (DAQ)****

- ADC Topology: Successive Approximation Register (SAR).
- ADC Resolution: 12 bits (4096 discrete levels).
- SNR: 60-70 dB (corresponding to a dynamic range of about 10-11 bits).
- Input Capacitance and Leakage Current: Low to prevent loading effects.
- Power Supply Voltage: Typically 3.3 V to 5 V for digital circuitry.
- Sampling Rate: 1000 samples per second.
- Maximum Input Voltage: 7 V.

****General Considerations****

- All Op-Amps should have decoupling capacitors to reduce noise.
- Protection diodes may be used at the buffer output to protect the DAQ from voltage spikes.
- A compact design using SMD components is preferred for modern applications, but through-hole components may be used for prototyping ease.
- The entire system must be calibrated and tested to ensure accurate angle measurement within the operating voltage and angle range.

This comprehensive solution ensures that the pendulum angle is accurately measured and conditioned for analog-to-digital conversion without exceeding the DAQ's input voltage limits or introducing significant noise or distortion to the signal.

Listing B.2 – Nota e justificativas gerados pelo GPT avaliador para o melhor resultado do *testbench* de posição angular.

Score: 7

Explanations:

The project summary addresses several of the required items for the electronics project:

1. The potentiometer is used as a voltage divider: The summary does not explicitly state that the potentiometer is used as a voltage divider, but it implies it by stating that the voltage varies linearly with the angle, which is typical for a voltage divider application.
2. The voltage applied to the potentiometer is +/- 10V: The power supply for the operational amplifiers is +/- 10V, which suggests that this is also the voltage applied to the potentiometer.
3. The architecture includes a voltage divider, an anti-aliasing filter, and DAQ measurement: The summary describes a voltage follower (buffer), a gain stage, a notch filter, a low-pass filter, and a DAQ, which meets the requirement for a simple architecture.
4. The input voltage of the DAQ is centered at 0, +/- 7V: The gain stage is designed to output a voltage range of +/- 7V, which matches the DAQ's maximum input voltage specification.
5. The maximum voltage applied to the DAQ is 7V: This is explicitly stated in the DAQ section as the maximum input voltage.
6. There is a low-pass filter that avoids aliasing: The anti-aliasing filter is a Sallen-Key low-pass filter with a cutoff frequency of approximately 210 Hz, which is suitable for the given sampling rate of 1000 samples per second.
7. There is a filter removing frequencies between around 50 and 60 Hz: The band-pass filter section describes an active twin-T notch filter topology specifically for attenuating 50 Hz and 60 Hz frequencies.
8. The filter has a cutoff frequency and order such that the gain of the signal is at least -20 dB at 500 Hz: The low-pass filter has a cutoff frequency of 200 Hz, which would result in a gain of less than -20 dB at 500 Hz due to the roll-off rate of -12 dB/octave.

All the requirements seem to be met based on the provided summary, except for the explicit mention of the potentiometer being used as a voltage divider, which is inferred rather than directly stated.

Listing B.3 – Veredito e justificativas gerados pelo GPT avaliador para o melhor resultado do *testbench* de posição angular.

Verdict: acceptable

Explanations:

The project summary provides a detailed outline for a pendulum angle measurement system, which includes various stages of signal conditioning and filtering before the data acquisition (DAQ) process. The following points assess the compliance with the given requirements:

1. The potentiometer is correctly used as a voltage divider, with its output expected to vary between -5V and +5V for the pendulum's angular range.
2. The power supply for the potentiometer and operational amplifiers is stated to be 10V, which aligns with the requirement.
3. The architecture is relatively simple, consisting of a buffer, gain stage, filters, and a DAQ system.
4. The input voltage of the DAQ is centered at 0V, with a range of 7V, meeting the specified requirement.
5. The maximum voltage applied to the DAQ is within the 7V limit, as ensured by the gain stage design.
6. There is a mention of an anti-aliasing filter with a second-order Sallen-Key low-pass topology and a cutoff frequency of approximately 210 Hz, which should help mitigate aliasing.
7. The band-pass filter includes a Twin-T notch filter to attenuate 50 Hz and 60 Hz, which would effectively remove unwanted frequencies from the power line interference.
8. The low-pass filter with a cutoff frequency of 200 Hz and a second-order response would have a gain of at least -20 dB at 500 Hz, as it would roll off at -12 dB/octave beyond the cutoff frequency.

The design appears to be well thought out and should be capable of meeting the project requirements. However, the summary does not explicitly mention the gain at 500 Hz for the anti-aliasing filter. Although it can be inferred that the filter would meet the -20 dB requirement at 500 Hz due to the filter order and cutoff frequency, this is not explicitly stated. Additionally, the gain stage calculation does not consider the effect of the voltage follower's finite gain and output impedance on the overall system gain, which might be important for precision.

Despite these minor concerns, the project summary suggests that the design is well within the acceptable range and does not contain fatal issues.

B.2 Laboratório de Termometria

Listing B.4 – Melhor resultado para o *testbench* de termometria, obtida na iteração 3 de contexto investigativo.

Analog Water Temperature Measurement System

The project entails developing an analog system that measures the temperature of water using a thermistor and outputs a voltage range of 0-20V to be measured by a multimeter.

NTC Thermistor:

- Type: Vishay NTCLE100E3 series, assumed to have a 10k ohm resistance at 25 degrees Celsius and a beta value of approximately 3950K for calculations.
- Waterproofing and environmental protection considerations should be observed based on application needs.

Linearization Resistor:

- To linearize the thermistor response at 50 degrees Celsius (assumed midpoint temperature), calculate the resistance at 50 degrees Celsius (R_{50}) using the beta value and then determine the parallel resistor (R_p) value.
- Example calculation for R_{50} and R_p is omitted as it requires the precise beta value and resistance at the desired temperature.

Wheatstone Bridge:

- Bridge Configuration: Ratiometric with an assumption of 5V excitation voltage.
- Output Voltage Range: Design for 0-100 mV differential voltage output from the bridge.
- Resistors: Use precision resistors of equal value, assumed to be 3.3k ohms each based on the midpoint resistance approximation of the thermistor.

Instrumentation Amplifier:

- Gain Calculation (assuming bridge output of 0-100mV for full temperature range): Aim for a gain of 200 to scale the maximum expected input voltage to 20V output.
- Example Gain Calculation:
$$G = V_{out_max} / V_{in_max}$$
$$G = 20V / 0.1V$$
$$G = 200$$
- Gain-setting Resistor (R_g):
$$R_g = 50\text{ k} / (G - 1)$$
$$R_g = 252.5 \text{ (use closest standard value, e.g., 255)}$$

Low-Pass Filter:

- Topology: Second-order active Butterworth filter for a flat passband.

- Roll-off Rate: 12 dB/octave (40 dB/decade).
- Cutoff Frequency: 1 Hz.
- Q Factor: 0.707 for critical damping.
- Components and values should be calculated based on op-amp characteristics and desired cutoff frequency.

Output Stage:

- Topology: Op-amp based voltage follower (buffer) with scaling gain to achieve a 0 to 20V output range.
- Gain-setting Resistors (R1 and R2):
$$R1 = R2 * (G - 1)$$
$$R1 = 1 \text{ k} * (200 - 1) = 199 \text{ k (use closest standard value, e.g., 200 k)}$$
$$R2 = 1 \text{ k}$$

Multimeter:

- Features: Digital multimeter with auto-ranging capability, high input impedance (1M or greater), capable of measuring up to 20V.
- Resolution: Typically 1 mV or better for temperature measurement applications.
- Usage: Ensure proper connection of probes observing the polarity of the DC voltage output.

The above solution provides a high-level design approach to the analog temperature measurement system. However, all calculations are based on assumptions and typical values. For a precise design, specific thermistor characteristics, including resistance-temperature relationship, beta value, and exact application requirements, need to be known. Also, the exact power supply voltages available for the system were not specified, and these would have an impact on the design and selection of components, particularly for the instrumentation amplifier and the low-pass filter.

Listing B.5 – Nota e justificativas gerados pelo GPT avaliador para o melhor resultado do *testbench* de termometria.

Score: 7

Explanations:

The project summary covers the following requirements:

1. The output is being measured by a multimeter: The summary explicitly states that a digital multimeter with auto-ranging capability is used to measure the output voltage range of 0-20V.

2. Amplification stage with gain provided and justified: The project describes using an instrumentation amplifier with a gain of 200, which is justified by the need to scale the maximum expected input voltage of 100mV to an output of 20V.
3. Output voltage range between 0 and 20 Volts: The project aims to achieve an output voltage range of 0-20V, which aligns with the requirement.
4. NTC linearized by a resistor optimized for the midrange 50 C: The summary indicates that the NTC thermistor is linearized at 50 degrees Celsius using a calculated parallel resistor, although the exact calculation is omitted.
5. NTC linearization: The summary reaffirms the importance of linearization and indicates that it is part of the design, even though the exact calculation is not provided.
6. Architecture: The project summary follows the required architecture, including the sensor (NTC thermistor), linearization stage, amplification, optional filtering (low-pass filter), and measurement.
7. Sensor used is an NTC thermistor: The project specifies the use of a Vishay NTCLE100E3 series NTC thermistor.
8. Self-heating effect is taken into account: Although the summary does not explicitly state the maximum current passing through the NTC, it mentions the consideration of waterproofing and environmental protection, which implies that self-heating might have been considered. However, without specific information on the maximum current, this requirement is not fully met.

The project does not provide details on the exact beta value, precise resistance-temperature relationship, or power supply voltages, which are necessary for a complete assessment of the design. Additionally, the maximum current through the NTC is not specified. Therefore, the score is based on the requirements that are clearly met.

Listing B.6 – Veredito e justificativas gerados pelo GPT avaliador para o melhor resultado do *testbench* de termometria.

Verdict: acceptable

Explanations:

The project summary provides a detailed approach to designing an analog water temperature measurement system using an NTC thermistor, which is in line with requirement 7. It includes linearization of the NTC thermistor at 50 degrees Celsius, which satisfies requirement 4. The architecture is suitable and consists of the sensor, linearization stage, amplification, and optional filtering which meets requirement 6. The gain of the amplification stage is provided and justified, fulfilling requirement 2. The output voltage range is designed to be within 0 to 20 Volts, which complies with requirement 3.

However, there are some concerns and omissions in the project summary:

- The summary does not explicitly mention how the self-heating effect is taken into account, nor does it specify the maximum current that passes through the NTC. This is a requirement marked as essential (requirement 8) and the absence of this information is a significant oversight.
- The linearity of the NTC is mentioned, but there is no specific explanation or evidence provided to confirm that the NTC is linearized, which is crucial as per requirement 5.
- The description of the low-pass filter and output stage is somewhat generic, and while it does not directly contravene any requirements, it does not add significant value to the specific application of temperature measurement.

Given these points, the project summary does not completely satisfy all the essential requirements, particularly the self-heating effect and the confirmation of the NTC linearization. Therefore, the verdict for this project summary is "acceptable" as the project can be implemented, does not have any fatal issues, but the score isn't perfect due to the omission of crucial information regarding self-heating and the lack of explicit evidence of NTC linearization.

B.3 Projeto de Acelerometria

Listing B.7 – Melhor resultado para o *testbench* de acelerometria, obtida na iteração 18 de contexto direto.

Portable Low-Frequency Vibration Measurement Device Design

Sensor:

- Selected Sensor: Dytran 3225M36 piezoelectric accelerometer with a sensitivity of 100 pC/g.
- Operating Temperature Range: -40C to +85C.
- Size and Weight: Compact and lightweight design suitable for portable devices.
- Shock and Vibration Resistance: Robust to withstand significant levels of shock and vibration.

Charge Amplifier:

- Gain (G) Calculation: To achieve a 1 Vpp output for a peak charge (Q_peak) of 162.28 pC, $G = 1 \text{ V} / 162.28 \text{ pC} = 6.16 \times 10^6 \text{ V/C}$.
- Feedback Capacitor (C_f): Chosen as 10 pF.
- Feedback Resistor (R_f): Selected as 68 M for a cutoff frequency of 0.25 Hz, $R_f = 1 / (2 \times 0.25 \text{ Hz} \times 10 \text{ pF}) = 63.66 \text{ M}$.
- Operational Amplifier: An ultra-low bias current op-amp like the LMC662 is used to minimize offset voltage.

Low Pass Filter:

- Topology: Active multiple-feedback design.
- Filter Type: 4th-order Butterworth with a cutoff frequency of 2 Hz.
- Op-Amp: AD8628 or equivalent with low input bias current, low offset voltage, and low noise.
- Component Values: Component values would be calculated based on standard Butterworth coefficients, within the range of 10 nF to 100 nF for capacitors to maintain practical resistor sizes.

Output Stage:

- Topology: Non-Inverting Op-Amp configuration.
- Supply Voltage: +5V for single-supply operation.
- Op-Amp Selection: Low-power, low-noise op-amps like Texas Instruments OPA333 or Analog Devices AD8605.
- Gain Adjustability: A potentiometer in the feedback loop for adjustable gain.
- Feedback Components: Rf and Cf values to match expected gain, with additional series resistor Rs for bias current balancing.

ADC:

- Type: 16-bit Sigma-Delta () ADC with a differential input.
- Sampling Rate: At least 10 SPS to capture the low-frequency signal with sufficient detail.
- Input Range: Set to match the 1 V peak-to-peak range from the charge amplifier, adjusted for the 10 mV offset.
- Interface: I2C or SPI for data transfer to the microcontroller.

- Differential Input: Preferred to minimize common-mode noise in a portable device environment.

Overall Considerations:

- The entire design is focused on ensuring the fidelity of the low-frequency vibration signal from the sensor through to the digital domain, with attention to power consumption for portability.
- The analog front-end maintains a high signal-to-noise ratio while the ADC provides high-resolution digital conversion.
- The system is designed to be robust against environmental factors and low power consumption for battery operation.

Listing B.8 – Nota e justificativas gerados pelo GPT avaliador para o melhor resultado do *testbench* de acelerometria.

Score: 5

Explanations:

1. The feedback resistance (R_f) and feedback capacitance (C_f) are given as 68 M and 10 pF, respectively. This gives an $R_f C_f$ product of $68 \text{ M} * 10 \text{ pF} = 680 \text{ s}$, which is not close to $2 / (0.637)$ with a 10 percent tolerance. This requirement is not met.
2. The charge amplifier gain (G) of $6.16 * 10^6 \text{ V/C}$ and the feedback capacitance (C_f) of 10 pF is used to check the second requirement. Multiplying the gain by $1.61\text{E-}12$ and dividing by the feedback capacitance gives $(6.16 * 10^6 \text{ V/C}) * (1.61\text{E-}12 \text{ C/pC}) / (10 \text{ pF}) = 0.98$, which is within the 10 percent tolerance of 1. This requirement is met.
3. The peak-to-peak charge output is stated as 162.28 pC, which is very close to the required 161 pC. This requirement is met.
4. The bias current is mentioned to be balanced with a series resistor in the feedback loop, but the actual value is not provided. This requirement is not explicitly met.
5. The feedback resistance (R_f) is given, but there is no mention of the bias current being 0.01 divided by the feedback resistance. This requirement is not explicitly met.
6. The project uses a charge amplifier to condition the signal from a piezoelectric sensor. This requirement is met.

7. The project includes a low-pass filter with a cutoff frequency of 2 Hz, which is optimized for the input oscillation of 2 Hz, but there is no explicit mention of calculations backing up the 5 cm amplitude. This requirement is not fully met.
8. The output voltage is designed to be 1 V peak to peak. This requirement is met.
9. An ultra-low bias current op-amp is used to minimize offset voltage, and the ADC input range is adjusted for the 10 mV offset, suggesting that the offset is kept below 10 mV. This requirement is met.

Overall, the project summary covers 5 out of the 9 requirements, as requirements 1, 4, 5, and 7 are not fully met based on the information provided.

Listing B.9 – Veredito e justificativas gerados pelo GPT avaliador para o melhor resultado do *testbench* de acelerometria.

Verdict: unfeasible

Explanations:

The project design for a Portable Low-Frequency Vibration Measurement Device has several good features, but there are discrepancies between the design parameters and the project requirements:

1. The feedback resistor (R_f) and feedback capacitance (C_f) do not meet the requirement of $R_f * C_f$ being roughly around $2/\pi$ (allowing up to 10 percent difference). Given $R_f = 68 \text{ M}$ and $C_f = 10 \text{ pF}$, $R_f * C_f = 680 \text{ ns}$, which is significantly higher than the target value of approximately 0.637 ns ($2/\pi = \sim 0.637$, allowing for 10 percent difference).
2. The gain calculation (G) multiplied by $1.61\text{E-}12$ and divided by the feedback capacitance ($G * 1.61\text{p} / C_f$) should be roughly around 1, allowing up to 10 percent difference. With $G = 6.16 * 10^6 \text{ V/C}$ and $C_f = 10 \text{ pF}$, the value $G * 1.61\text{E-}12 / C_f = 0.984$, which is within the acceptable range.
3. The peak-to-peak charge output is calculated to be around 162.28 pC , which is close to the required 161 pC .
4. Bias current is mentioned, but the specific value of the bias current is not provided. The requirement for the bias current to be provided and for

0.01 divided by the feedback resistance to equal the bias current is not explicitly met in the provided information.

5. The project uses a charge amplifier to condition the piezoelectric sensor, which aligns with the requirements.
6. The design is optimized for an input oscillation of 2 Hz, which is in line with the requirement. However, there is no mention of calculations backing up the optimization for a 5 cm amplitude.
7. The output voltage is specified to be 1 V peak to peak, which meets the requirement.
8. The use of an ultra-low bias current op-amp is mentioned to minimize offset voltage, but there is no explicit confirmation that the offset is kept below 10 mV.

In conclusion, the design does not fully comply with the essential requirements, specifically in the feedback network ($R_f * C_f$ value) and the provision of bias current details. Additionally, there are missing calculations for the optimization of the input oscillation amplitude and no explicit confirmation of the offset voltage.

B.4 Projeto de Pressão e Temperatura Superficial

Listing B.10 – Melhor resultado para o *testbench* de pressão e temperatura superficial, obtida na iteração 8 de contexto direto.

Pressure and Temperature Monitoring System Design

1. Pressure Sensor Array

- Pressure Range: 0 to 10 bar.
- Sensitivity: Assuming 2 mV/V/bar with a 5V excitation voltage.
- Accuracy: 1 percent full scale (FS), equating to a maximum error of 0.1 bar at full scale.
- Sensor Model Suggestion: Honeywell S&C - MLH Series.
- Configuration: Wheatstone bridge.
- Operating Conditions: Industrial-grade sensors to handle -40 to 85C, suitable for humidity and corrosive substances.

2. Temperature Sensor Array

- Temperature Range: -40 to 85C (typical industrial range).
- Resolution: 0.1C is assumed as adequate.
- Sensor Model Suggestion: MLX90614 by Melexis. Digital output with a resolution of 0.02C.
- Operating Conditions: Sensor to operate in humidity and exposure to chemicals.
- Supply Voltage: Standard industrial supply of 24 V DC, with sensors operating at 3.3 or 5 V with a voltage regulator.

3. Instrumentation Amplifiers for Pressure Sensors

- Gain: To match the ADC's 0-5V range, a gain of 5000 is required.
- Operational Amplifiers: Precision op-amps with CMRR > 100 dB are chosen.
- Power Supply: Standard 15V dual supply for amplifiers.
- Input Range: 0 to 1 mV from the pressure sensors.
- Configuration: Three-op-amp instrumentation amplifier topology.
- Resistors: $R_1 = 1 \text{ k}$, R_{gain} calculated to be approximately 400.08 for the gain of 5000.

4. Signal Conditioner (Nonlinearity Correction) for Temperature Sensors

- Correction Method: Polynomial correction circuit with operational amplifier stages configured to generate a polynomial function for sensor nonlinearity.
- Operational Amplifiers: Low bias current and low offset voltage op-amps like LT1013 or AD712.
- Resistors and Potentiometers: Precision 1 percent metal film resistors and high-resolution potentiometers for calibration.

5. Voltage Amplifiers for Temperature Sensors

- Gain: 100 to match the ADC's 0-5V range from a maximum sensor output of 100 mV.
- Operational Amplifiers: Low-noise, high slew-rate op-amps like TL072 or equivalent.
- Resistors: 99 k feedback and 1 k input for setting the gain.
- Power Supply: Dual supply (e.g., 15V) to support the required output swing.

6. Multiplexer

- Model: CD74HC4067 or equivalent 16-channel analog multiplexer.
- Bandwidth: >4 kHz.
- Settling Time: <100 s.
- On-Resistance: <100 ohms.
- Voltage Range: At least 0-5V.

- Control Interface: Digital, compatible with logic levels of system microcontroller/processor.
7. Anti-Aliasing Filter
- Type: Sallen-Key Butterworth Low-Pass Filter.
 - Order: 4th Order (Two 2nd Order Stages).
 - Cutoff Frequency: 400 Hz.
 - Stopband Attenuation: At least 48 dB by the first octave above cutoff (800 Hz).
 - Operational Amplifiers: TL072 or equivalent.
 - Resistors and Capacitors: Calculated based on the cutoff frequency of 400 Hz, with exact values fine-tuned during the design phase.
8. Analog-to-Digital Converter (ADC)
- Type: Successive Approximation Register (SAR) ADC.
 - Resolution: 12-bit.
 - Sampling Rate: At least 2 kHz per channel.
 - Input Voltage Range: 0-5V or higher.
 - Anti-Aliasing Filter: Integrated if available, otherwise external with a cutoff frequency above 400 Hz.
9. Data Interface
- Communication Protocol: USB 2.0 standard.
 - Data Rate: 150 kbps required (12 bits/sample * 800 samples/second * 16 channels).
 - Interface Components: USB microcontroller with FIFO buffer, optional galvanic isolation, and power regulation.
-

Listing B.11 – Nota e justificativas gerados pelo GPT avaliador para o melhor resultado do *testbench* de pressão e temperatura superficial.

Score: 11

Explanations:

The project summary covers the following requirements:

1. Both sensors have d.c. output: The pressure sensor outputs a millivolt level signal, which is DC. The temperature sensor (MLX90614) has a digital output, which implies a DC level signal (though digital). Hence, this requirement is met.
2. Both sensors must be amplified: The pressure sensor is amplified by an instrumentation amplifier, and the temperature sensor signal is amplified by a voltage amplifier. This requirement is met.

3. The pressure sensor is inserted in a Wheatstone bridge and amplified by an instrumentation amplifier: This is explicitly stated in the summary. This requirement is met.
4. An ADC is used: A 12-bit SAR ADC with a sampling rate of at least 2 kHz per channel is mentioned. This requirement is met.
5. Infrared radiation sensors are being linearized: Polynomial correction circuit with operational amplifier stages is used for nonlinearity correction. While this doesn't explicitly mention linearizing infrared sensors, the MLX90614 is an infrared sensor, and the polynomial correction is a valid linearization technique. This requirement is met.
6. The solution mentions the sampling order strategy: There is no explicit mention of the sampling order strategy (sequential, simultaneous, etc.). This requirement is not met.
7. The sampling frequency of the ADC is not less than 800 Hz: The ADC has a sampling rate of at least 2 kHz per channel, which exceeds 800 Hz. This requirement is met.
8. The anti-aliasing filter has a cutoff frequency and order such that the gain of the signal is of at least -20 dB at half the sampling frequency: The stopband attenuation is at least 48 dB by the first octave above the cutoff (800 Hz), which is more than -20 dB at half the sampling frequency of 1 kHz (half of 2 kHz). This requirement is met.
9. The low pass cutoff frequency is higher than 400 Hz and lower than half the total sampling frequency: The cutoff frequency of the anti-aliasing filter is 400 Hz, which meets the first part of this requirement. Since the sampling frequency is at least 2 kHz, half of this is 1 kHz, so the cutoff frequency is also lower than half the total sampling frequency. This requirement is met.
10. There are low-pass filters (or anti-aliasing filters) to reduce aliasing, which are positioned before the multiplexer(s): The presence of anti-aliasing filters is mentioned, but their exact positioning relative to the multiplexer is not detailed. However, since the filters are part of the signal conditioning, it can be reasonably inferred that they are positioned before the multiplexer, which is typical in such designs. This requirement is met implicitly.
11. The project uses multiplexers to choose channels: A 16-channel analog multiplexer (CD74HC4067) is used. This requirement is met.
12. The multiplexers are solid state: The CD74HC4067 is a solid-state device. This requirement is met.

The requirements that are not met are:

- Requirement 6: The sampling order strategy is not mentioned.

Therefore, the project successfully meets 11 out of 12 requirements.

Listing B.12 – Veredito e justificativas gerados pelo GPT avaliador para o melhor resultado do *testbench* de pressão e temperatura superficial.

Verdict: acceptable

Explanations:

The project summary provided for the Pressure and Temperature Monitoring System Design outlines a comprehensive system that includes pressure and temperature sensors, amplification stages, multiplexing, signal conditioning, anti-aliasing filtering, and analog-to-digital conversion, meeting many of the requirements specified. However, there are some issues to address:

1. Both sensors have d.c. output, and no demodulation is used this requirement is met.
2. Both sensors are amplified this requirement is met with the use of instrumentation amplifiers for pressure sensors and voltage amplifiers for temperature sensors.
3. The pressure sensor is in a Wheatstone bridge and amplified by an instrumentation amplifier this requirement is met.
4. An ADC is used this requirement is met with a specified SAR ADC with a resolution of 12-bit.
5. Infrared radiation sensors are being linearized this requirement is met with the implementation of a polynomial correction circuit for the temperature sensors.
6. The sampling order strategy is not explicitly mentioned this requirement is not fully met as the summary does not specify whether the sampling is sequential or simultaneous.
7. The ADC's sampling frequency is at least 800 Hz this requirement is met, with a specified sampling rate of at least 2 kHz per channel.
8. The anti-aliasing filter's specifications meet the requirement of at least -20 dB at half the sampling frequency this requirement is met with a stopband attenuation of at least 48 dB by the first octave above the cutoff (800 Hz).
9. The low pass cutoff frequency is between 400 Hz and less than half the total sampling frequency this requirement is met, as the cutoff frequency is specified at 400 Hz.
10. Low-pass filters are positioned before the multiplexer(s) this requirement is met as the anti-aliasing filter is mentioned, though the exact positioning is not specified.

11. Multiplexer(s) are used this requirement is met with the inclusion of a 16-channel analog multiplexer.
12. Multiplexers are solid-state this requirement is met with the specified CD74HC4067.

The only minor shortfall is the lack of detail on the sampling order strategy; however, this does not constitute a fatal flaw and can be easily addressed in the implementation phase. Overall, the project meets all essential requirements and can be considered well thought out and applicable.

APÊNDICE C – Tabela Padrão de U

Tabela 23 – Valores críticos de U para $\alpha = 0,05$.

-	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	-	-	-	-	-	-	0	0	0	0	1	1	1	1	1	2	2	2	2
3	-	-	-	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8
4	-	-	0	1	2	3	4	4	5	6	7	8	9	10	11	11	12	13	14
5	-	0	1	2	3	5	6	7	8	9	11	12	13	14	15	17	18	19	20
6	-	1	2	3	5	6	7	10	11	13	14	16	17	19	21	22	24	25	27
7	-	1	3	5	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
8	0	2	4	6	7	10	13	15	17	19	22	24	26	29	31	34	36	38	41
9	0	2	4	7	10	12	15	17	20	23	26	28	31	34	37	39	42	45	48
10	0	3	5	8	11	14	17	20	23	26	29	33	36	39	42	45	48	52	55
11	0	3	6	9	13	16	19	23	26	30	33	37	40	44	47	51	55	58	62
12	1	4	7	11	14	18	22	26	29	33	37	41	45	49	53	57	61	65	69
13	1	4	8	12	16	20	24	28	33	37	41	45	50	54	59	63	67	72	76
14	1	5	9	13	17	22	26	31	36	40	45	50	55	59	64	67	74	78	83
15	1	5	10	14	19	24	29	34	39	44	49	54	59	64	70	75	80	85	90
16	1	6	11	15	21	26	31	37	42	47	53	59	64	70	75	81	86	92	98
17	2	6	11	17	22	28	34	39	45	51	57	63	67	75	81	87	93	99	105
18	2	7	12	18	24	30	36	42	48	55	61	67	74	80	86	93	99	106	112
19	2	7	13	19	25	32	38	45	52	58	65	72	78	85	92	99	106	113	119
20	2	8	13	20	27	34	41	48	55	62	69	76	83	90	98	105	112	119	127

Fonte: Sani e Todman (2006).