UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE ENGENHARIA ELÉTRICA

RAFAEL WAZLAWICK MÜLLER

DESENVOLVIMENTO DE SISTEMA FUNCIONAL DE TESTES EM TEMPO REAL COM IHM PARA VALIDAÇÃO DE CONTROLADORES AUTOMOTIVOS

São Leopoldo
2020

RAFAEL WAZLAWICK MÜLLER

DESENVOLVIMENTO DE SISTEMAS FUNCIONAL DE TESTES EM TEMPO REAL COM IHM PARA VALIDAÇÃO DE CONTROLADORES AUTOMOTIVOS

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica, pelo Curso de Engenharia Elétrica da Universidade do Vale do Rio dos Sinos - UNISINOS

Orientador: Prof. Ms. João Olegário de Oliveira de Souza

São Leopoldo

2020

RESUMO

O projeto em questão, teve por objetivo propor e desenvolver uma solução para teste e validação de produtos para aplicações automotivas, com base em sistemas comerciais existentes, de modo a eliminar a necessidade de módulos de teste exclusivos para cada controlador.

Para tanto, foram estudadas diversas topologias de hardware destinadas ao desenvolvimento de pequenos módulos de teste que compunham uma rede, que de forma conjunta permitam validar e testar os produtos de modo geral. Estes circuitos tratam de aquisição e simulação de dados, isto é, emulação de sensores de temperatura, leitura de entradas e saídas digitais, analógicas e PWM (*Pulse Width Modulation,* ou, em português, Modulação por Largura de Pulso.

Em seguida, após obtenção e geração de dados, estas informações são enviadas e comandadas via rede de comunicação a uma IHM (Interface Homem-Máquina) no qual possibilite ao usuário realizar a leitura e manipulação do teste a ser realizado.

De modo geral, o projeto atingiu os resultados esperados, sendo possível realizar todos os controles e configurações propostas. A IHM possibilita o cadastro de um novo teste e o hardware se adapta conforme esta configuração. Entretanto, à medida que a validação foi realizada, foram verificadas possíveis melhorias para implementações futuras.

Palavras-chave: Teste e validação. Aquisição e simulação de dados. Simulação. IHM.

LISTA DE FIGURAS

Figura 1 – Sistema funcional de testes 11
Figura 2 – Diagrama de blocos do sistema 14
Figura 3 – TS-3000 17
Figura 4 – ATS 420X 18
Figura 5 – Exemplo 1 de aplicação no QT 19
Figura 6 – Exemplo 2 de aplicação no QT 19
Figura 7 – Cadeia de resistores 20
Figura 8 – Multímetros Digitais 21
Figura 9 – Módulo E/S 22
Figura 10 – Placa de desenvolvimento STM32F103C8 26
Figura 11 – Interface Serial RS-232 27
Figura 12 – Interface I2C 28
Figura 13 – Escrita Barramento I2C 29
Figura 14 – Diagrama de blocos de uma DAQ 30
Figura 15 – Conexão típica de um conversor A/D 32
Figura 16 – Filtro RLC 32
Figura 17 – Topologias de saídas 33
Figura 18 – Diagrama de funcionamento de potenciômetro digital 34
Figura 19 – Raspberry Pi 4 Model B 35
Figura 20 – Diagrama de blocos de funcionamento do protótipo. 37
Figura 21 – Fonte de alimentação do protótipo 38
Figura 22 – Tela inicial 39
Figura 23 – Tela de cadastro de novo teste. 40
Figura 24 – Tela de seleção de produto 41
Figura 25 – Tela de teste de produto em branco. 42
Figura 26 – Tela de carregamento de arquivo. 43
Figura 27 – Exemplo saída digital positiva 43
Figura 28 – Exemplo saída digital negativa 44
Figura 29 – Exemplo saída digital PWM 44
Figura 30 – Exemplo saída analógica 44
Figura 31 – Exemplo entrada digital 45
Figura 32 – Exemplo entrada PWM 45

Figura 33 – Exemplo entrada analógica....45
Figura 34 – Exemplo sensor de temperatura 46
Figura 35 – Implementação serial módulos....47
Figura 36 – Conexão porta serial QT Creator 47
Figura 37 – Mensagem serial....49
Figura 38 – Redistribuição da mensagem serial 50
Figura 39 – CRC recebido....51
Figura 40 – Tratamento de dados 51
Figura 41 – Verificação de erro CRC 52
Figura 42 – Diagrama de ligação dos potenciômetros digitais 53
Figura 43 – Minipa MDM-8156B....54
Figura 44 – Equação da reta característica 150 Ω a 1000 Ω. 55
Figura 45 – Byte de Endereços AD5241 e AD5175 56
Figura 46 – Fluxograma software módulo de resistências 57
Figura 47 – Inicialização AD5241....58
Figura 48 – Inicialização AD5175....58
Figura 49 – Tabela de coeficientes 58
Figura 50 – Código principal Módulo de resistências....59
Figura 51 – Ajuste de posição AD5241 1 MΩ 60
Figura 52 – Ajuste de posição 10 kΩ 61
Figura 53 – Esquemático módulo de resistências....62
Figura 54 – Hardware desenvolvido Módulo de resistências 62
Figura 55 – Fluxograma software módulo de entradas 63
Figura 56 – Configuração TI1FP1 64
Figura 57 – Código Etapa PWM....65
Figura 58 – Código Etapa Digital / Analógico.....66
Figura 59 – Parâmetros VREFINT 67
Figura 60 – Esquemático módulo de entradas....68
Figura 61 – Hardware desenvolvido Módulo de entradas 68
Figura 62 – Fluxograma software Módulo de saídas Bloco 1....69
Figura 63 – Fluxograma software Módulo de saídas Bloco 2....70
Figura 64 – Código saída PWM / digital....74
Figura 65 – Circuito Saída PWM / Digital....75
Figura 66 – Circuito Saída analógica 77

Figura 67 – Código saída analógica....78
Figura 68 – Ajuste A/D saída analógica....79
Figura 69 – Hardware desenvolvido Módulo de saídas....80
Figura 70 – Ajustes de temperatura na câmara térmica....83
Figura 71 – Potenciômetros digitais inseridos na câmara térmica....84
Figura 72 – Resultado das entradas PWM....85
Figura 73 – Resultado das entradas analógicas....86
Figura 74 – Resultado das entradas digitais....87
Figura 75 – Resultado das Saídas PWM....88
Figura 76 – Resultado das Saídas analógicas....89
Figura 77 – Ondulação saída analógica....90
Figura 78 – Resultado das Saídas digitais....90
Figura 79 – Setup completo....91
Figura 80 – Tela de cadastro de teste completa....103
Figura 81 – Tela de teste completa....104
Figura 82 – Filtro passa-baixas passivo....105
Figura 83 – Amplificador não-inversor....107

LISTA DE TABELAS

Tabela 1 – Comparação Arduino x Nucleo.....25
Tabela 2 – Determinação experimental 150 Ω a 1000 Ω. 54
Tabela 3 – Lógica para ADDR AD5175.....56
Tabela 4 – Definições de frequências 72
Tabela 5 – Erros varredura de frequências 72
Tabela 6 – Resultado entradas de resistência 82
Tabela 7 – Custos protótipo 92

LISTA DE SIGLAS

A/D	Analógico/Digital
ADC	*Analog Digital Converter* (Conversor Analógico Digital)
ARM	*Advanced Risc Machine (*Máquina RISC Avançada)
ATX	*Advanced Technology eXtended* (Tecnologia Avançada eXtendida)
BJT	*Bipolar Junction Transistor* (Transistor Bipolar de Junção)
CAN	*Controller Area Network* (Rede de área do controlador)
CPU	*Central Process Unit* (Unidade de Processamento Central)
CRC	*Cyclic Redundancy Check* (Verificação cíclica de redundância)
CS	*Chip Select* (Seleção de Chip)
DAQ	*Data Acquisition (*Aquisição de dados)
DMA	*Direct Memory Access* (Acesso de memória direta)
DMM	*Digital Multimeter* (Multímetro Digital)
DUT	*Device Under Test* (Dispositivo em teste)
E/S	Entradas/Saídas
ECU	Unidade de controlador eletrônico
ESD	*Electrostatic Discharge* (Descarga eletrostática)
FPGA	*Field Programmable Gate Array* (Arranjo de Portas Programáveis em Campo)
GND	*Ground* (Terra)
GUI	*Graphical User Interface* (Interface Gráfica do Usuário)
HDMI	*High-Definition Multimedia Interface* (Interface de multimídia em Alta Definição)
HVAC	(*Heating Ventilation and Air* Conditioning, ou, do português, Aquecimento, Ventilação e Ar-condicionado)
I2C	*Inter-integrated circuit (*Circuito interintegrado)
IDE	*Integrated Development Environment* (Ambiente de desenvolvimento integrado)
IHM	Interface Homem-Máquina
MOSFET	*Metal-Oxide-Semiconductor Field-Effect Transistor (*Transistor de Efeito de Campo de Óxido de Metal Semicondutor)
NTC	*Negative Temperature Coefficient* (Coeficiente de temperatura negativo)

P&D	Pesquisa e Desenvolvimento
PIC	*Programmable Interface Controller* (Controlador de Interface programável)
PTC	*Positive Temperature Coefficient* (Coeficiente de temperatura positivo)
PTH	*Pin Through Hole* (Pino através do buraco)
PWM	*Pulse Width Modulation (*Modulação por Largura de Pulso)
R/W	*Read/Write* (Leitura/Escrita)
RAM	*Random Access Memory* (Memória de Acesso Randômico)
RISC	*Reduced Instruction Set Computer (*Computador de Conjunto reduzido de instruções)
RTD	*Resistance Temperature Detector* (Detector de termorresistência)
SCL	*Serial Clock (*Clock Serial)
SCLK	*Serial Clock (*Clock Serial)
SDA	*Serial Data (*Dados seriais)
SMD	*Surface-mount Technology (*Tecnologia de Montagem Superficial)
SPDT	*Single pole Double Throw* (Polo único, acionamento duplo)
SPI	*Serial Peripheral Interface* (Interface de Periférico Serial)
USB	*Universal Serial Bus* (Porta Universal)
USART	*Universal Synchronous Asynchronous Receiver Transmitter (*Transmissor/Receptor Universal Síncrono e Assíncrono)
VCC	*Voltage Common Collector* (Tensão de Coletor Comum)
VI	*Virtual Instrument* (Instrumento Virtual)

SUMÁRIO

1 INTRODUÇÃO 11
1.1 Tema 13
1.2 Delimitação do Tema 14
1.3 Objetivos 14
1.3.1 Objetivo Geral 14
1.3.2 Objetivos Específicos 14
1.4 Justificativa 15
2 FUNDAMENTAÇÃO TEÓRICA 16
2.1 Testes funcionais 16
2.2 Sistemas comerciais 16
2.2.1 TS-3000 Sistema Funcional de testes 17
2.2.2 ATS 420X Sistema Funcional de testes 17
2.3 QT Creator 18
2.4 Módulos comerciais 20
2.4.1 Módulo de resistor programável Pickering 20
2.4.2 Simulador de termistores NTC Wireflow 21
2.4.3 Multímetros Digitais 21
2.4.4 Módulos de entradas e saídas 22
2.5 Microcontroladores 23
2.6 Kits de prototipagem e desenvolvimento 24
2.6.1 Arduíno 24
2.6.2 Nucleo STMicroeletronics 24
2.7 Protocolo de comunicação e detecção de erros 26
2.7.1 Protocolo serial 26
2.7.2 Protocolo I2C 27
2.7.3 CRC 30
2.8 Aquisição de dados 30
2.9 Técnicas para aquisição de dados 31
2.9.1 Medição de tensão 31
2.9.2 Topologias de Saídas 33
2.9.3 Potenciômetro digital 33
2.10 Microcomputadores 34

3 METODOLOGIA 36
3.1 Definições de hardware e software 37
3.2 Desenvolvimento IHM 38
3.3 Protocolo de comunicação 46
3.3.1 Configuração serial 46
3.3.2 Posicionamento de dados seriais 48
3.3.3 CRC 51
3.4 Módulo de resistências 52
3.5 Módulo de entradas PWM/Digitais/Analógicas 63
3.5.1 Etapa PWM 64
3.5.2 Etapa Digital / Analógica 66
3.6 Módulo de saídas PWM/Digitais/Analógicas 69
3.6.1 Etapa PWM / Digital 70
3.6.2 Etapa Analógica 76
4 RESULTADOS E DISCUSSÕES 81
4.1 IHM 81
4.2 Módulo de resistências 81
4.2.1 Medições 81
4.2.2 Influência da temperatura 82
4.3 Módulo de entradas 84
4.4 Módulo de saídas 87
4.4 Validação 91
4.5 Custos envolvidos 92
5 CONCLUSÃO 93
REFERÊNCIAS 95
APÊNDICE A – TELA DE CADASTRO DE NOVO TESTE 103
APÊNDICE B – TELA DE TESTE CARREGADA 104
APÊNDICE C – CÁLCULO DE FILTRO PASSA-BAIXAS 105
APÊNDICE D – CÁLCULO AMPLIFICADOR NÃO-INVERSOR 107

1 INTRODUÇÃO

Empresas que atuam na área de P&D (Pesquisa e Desenvolvimento), especificamente no desenvolvimento de novos produtos, necessitam de meios confiáveis para validação dos produtos a fim de promover confiabilidade e segurança na operação e funcionamento de seus equipamentos. Mediante esta informação, indústrias eletroeletrônicas empregam diferentes técnicas para o teste de seus produtos, podendo abranger métodos manuais e métodos totalmente automatizados que variam conforme a necessidade, aplicação ou o custo envolvido no processo.

Neste meio, existem dispositivos dedicados a testes de produtos que, basicamente, criam uma entrada para determinado caso, gerada por interação do usuário e hardware dedicado, e comparam as respostas obtidas (saídas reais do sistema) às previstas pelo teste. Esses sistemas são denominados sistemas funcionais de teste. A Figura 1 apresenta um típico dispositivo dedicado a este tipo de teste.

Figura 1 – Sistema funcional de testes

Fonte: (KEYSIGHT TECHNOLOGIES, 2019)

Em um cenário ideal são desejáveis testes automatizados, os quais, ainda que impliquem em um maior investimento, compensam no tempo de processo utilizado, principalmente quando comparados ao que é obtido por meio de um método de ~~um~~ teste manual. Entretanto, métodos como estes se tornam inviáveis para empresas que possuem uma vasta linha de produtos com diferentes funcionalidades e aplicações, dado que, por esse motivo, demandam um teste específico para cada novo produto desenvolvido, não havendo a possibilidade de recorrer a um teste “generalizado”.

Dessa forma, o presente trabalho buscou propor uma solução para teste de produtos, tendo como foco o desenvolvimento de um protótipo com base em sistemas comerciais conhecidos, apresentando interface com o usuário e hardware dedicado. Conforme necessário, o sistema pode se adaptar a diferentes configurações de produto. A validação se deu por meio de produtos de uma empresa já existente do ramo automotivo (adequando o projeto aos processos internos da empresa), que apresenta como seu diferencial a customização de controladores. Isto é, cada novo produto possui diferentes topologias e configurações.

1.1 Tema

A proposta alicerça-se no desenvolvimento de um módulo de teste genérico que tenha a capacidade de prever grande parte das necessidades de teste de controladores automotivos, resultando na dispensabilidade dos módulos atuais e da confecção de novos.

Controladores automotivos em geral, basicamente operam com entradas que controlam determinadas saídas de acordo com funções especificadas pelo cliente. Toma-se como exemplo as entradas de pressostatos, sensores de temperatura, e saídas como compressores, aquecedores e ventiladores, que normalmente são apresentados nas configurações digital, PWM ou analógico.

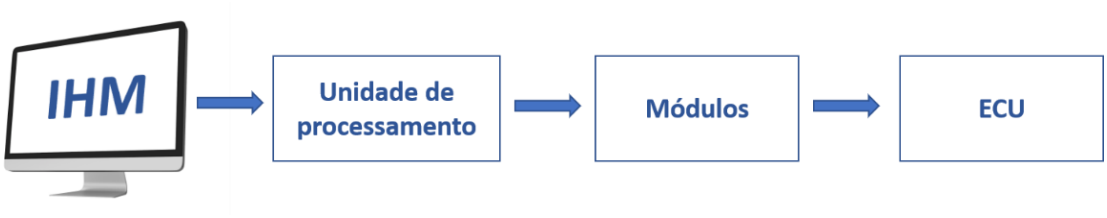
O tema do trabalho é idealizado através da implementação de sub-módulos dedicados a cada etapa de um determinado teste. Tais sub-módulos comportam uma rede de comunicação e são intercambiáveis, fornecendo flexibilidade a cada teste a ser realizado. Além disso, contribuem para testes futuros e desenvolvimento de novos sub-módulos, que possibilitam novas combinações.

Deste modo, para cada novo produto desenvolvido, devem ser avaliados os testes necessários, combinando os sub-módulos para montagem da estrutura do teste final. Para tanto, ante uma análise preliminar e com base nas necessidades da empresa alvo e seus produtos, foram desenvolvidos os seguintes sub-módulos para o protótipo inicial:

a) Módulo de saídas PWM, digitais e analógicas: saídas PWM podem ser ajustadas para frequências de 1 Hz a 100 kHz, saídas digitais possibilitam os estados de *on, off* e circuito aberto, e saídas analógicas ajustam uma tensão em uma faixa de 0 V a 10 V.
b) Módulo de entradas PWM, digitais e analógicas: permitem medições de até 100 kHz para entradas PWM e são capazes capturar o nível de tensão para entradas analógicas e digitais.
c) Módulo de resistências: consegue ajustar resistências de 150 Ω a 1 MΩ.

Em um primeiro momento, a IHM apresenta os dados em tempo real. Os dados são recebidos e transmitidos através dos módulos e, posteriormente, processados e interpretados. Os módulos permitem conexão com a *ECU* (*Electronic Controller Unit,* ou, do português, Unidade de Controlador Eletrônico). Assim, podemos esboçar na forma de diagrama a proposta elaborada, conforme mostrado na Figura 2.

Figura 2 – Diagrama de blocos do sistema



Fonte: Elaborado pelo autor.

1.2 Delimitação do Tema

O projeto consistiu no desenvolvimento dos módulos especificados dentre os propostos anteriormente. Cabe ressaltar que o protótipo não contempla um módulo de comunicações cujo objetivo seria o de validar as linhas de comunicação de um determinado produto. Isto se deve ao fato de o produto exigir um planejamento prévio, contemplando na sua etapa de projeto algum bloco dedicado a este tipo de teste.

Além disso, de um modo geral, o protótipo contempla somente a etapa de aceitação de teste, excluindo momentaneamente outras etapas que existem dentro do ambiente de um teste funcional completo.

1.3 Objetivos

1.3.1 Objetivo Geral

O presente trabalho tem por objetivo apresentar uma solução generalizada para testes de produtos de empresas, apresentando o desenvolvimento de hardware e software integrados.

1.3.2 Objetivos Específicos

Os objetivos específicos do projeto constam nos itens apresentados a seguir:

a) Desenvolver e construir um protótipo de módulos de testes que seja capaz de atender diversas aplicações dentro do conceito de teste de aceitação.
b) Promover um recurso que torne o teste de produtos mais eficiente, prático e dinâmico, e de custo inferior aos sistemas comerciais existentes.

c) Validar produtos eletrônicos com maior confiabilidade.
d) Verificar novos métodos e topologias de circuitos eletrônicos para testes.

1.4 Justificativa

O projeto utilizou, conforme mencionado anteriormente, uma empresa para validação do protótipo construído. Esta empresa, como diversas outras, confecciona manualmente módulos de teste específicos para cada novo produto, o que se deve especificamente pelo fato de a empresa trabalhar com o projeto de controladores customizáveis, resultando em um grande volume de produtos e havendo a demanda do desenvolvimento de módulos em grande quantidade.

.

2 FUNDAMENTAÇÃO TEÓRICA

No presente item serão apresentados os conceitos, técnicas e demais conteúdos que constituíram e fundamentaram o trabalho. Todas as definições foram realizadas com base nos tópicos abordados a seguir, que partem da escolha dos processadores para os módulos, forma de comunicação entre estes, topologias de circuitos para testes específicos, formas de acionamento de saídas, leitura das entradas e proteções elétricas, projeto da IHM e hardware em que esta irá operar, entre outros elementos.

Para tal, foram retratados sistemas comerciais que contribuíram para o desenvolvimento do protótipo, seguido das especificações gerais de projeto, tais como: as definições de processadores, os protocolos de comunicação e as placas de desenvolvimento. Nos tópicos posteriores, abordaram-se as estruturas dos módulos, no intuito de manter as subdivisões em ordem coerente.

2.1 Testes funcionais

Testes funcionais colocam a prova o funcionamento do produto x especificações realizadas pelo cliente. Também é designado como teste *black box* (do português, caixa preta) por não lidar com a verificação de código e outros fatores internos, e, sim, com o teste superficial de funcionamento. Dentro deste processo, existem diversas fases de teste.

Para o desenvolvimento do protótipo em questão, foi empregado o teste de aceitação. (AGTIC UFPR). Testes de aceitação são utilizados pela indústria e, de modo geral, possuem a função de determinar se as funcionalidades de um produto estão operando corretamente. (MILLER e COLLINS, 2002)

2.2 Sistemas comerciais

Esta seção apresenta módulos e componentes utilizados comercialmente para validação e teste de produtos.

2.2.1 TS-3000 Sistema Funcional de testes

O sistema apresentado na Figura 3 trata de um equipamento robusto com finalidade de efetuar testes funcionais, comercializado pela Keysight Technologies. Possui um computador integrado com software dedicado a realização de testes e conta com diversos canais de medições, entradas e saídas, e outros recursos. (KEYSIGHT TECHNOLOGIES, 2019).

Figura 3 – TS-3000

Fonte: (KEYSIGHT TECHNOLOGIES, 2019)

2.2.2 ATS 420X Sistema Funcional de testes

O sistema apresentado na Figura 4 trata de um sistema funcional de teste dedicado a indústria aeronáutica. Possui hardware da National Instruments, comunicação de alta velocidade e ferramentas de software próprias.

Figura 4 – ATS 420X

Fonte: (SET GMBH, 2019)

2.3 QT Creator

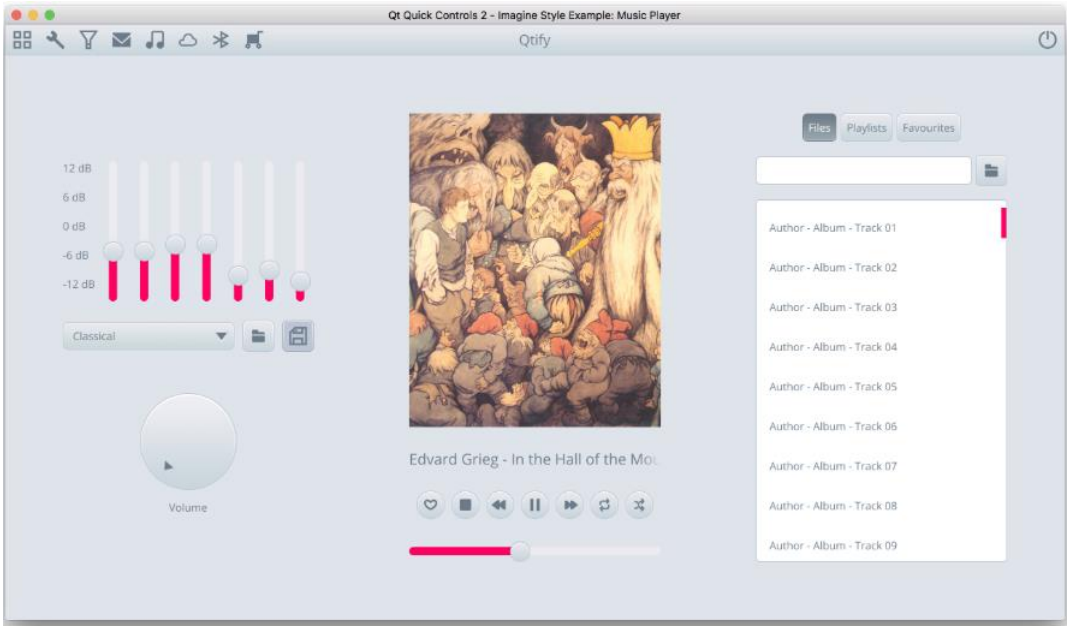
O QT é uma plataforma *open source* de desenvolvimento para aplicações GUI (*Graphical User Interface* ou, em português, Interface gráfica do usuário) diversas: teste, monitoramento, aquisição de dados, aplicativos de celular, dentre outros), permitindo, portanto, que os sistemas interajam com o usuário, semelhantemente a sistemas operacionais conhecidos como Windows, Android e iOS. (LAZAR e PENEA, 2016).

Abaixo, na Figura 5 e na Figura 6, são apresentados alguns exemplos de programas realizados através da plataforma QT.

Figura 5 – Exemplo 1 de aplicação no QT

Fonte: (QT GROUP, 2019)

Figura 6 – Exemplo 2 de aplicação no QT



Fonte: (QT GROUP, 2019)

O QT oferece ainda a possibilidade do uso da linguagem de programação C++. (LAZAR e PENEA, 2016).

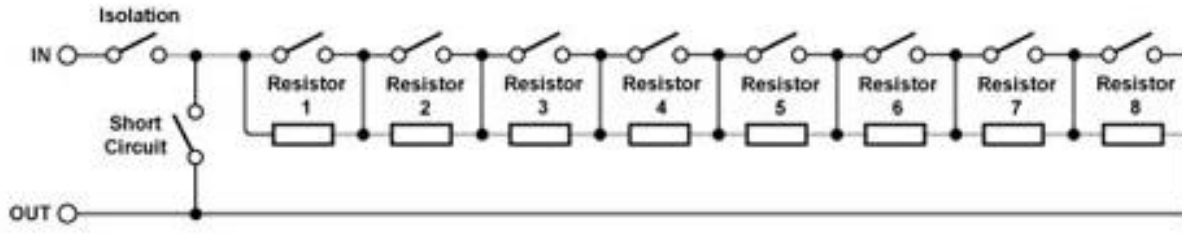
2.4 Módulos comerciais

2.4.1 Módulo de resistor programável Pickering

A Pickering Interfaces é uma empresa que desenvolve dispositivos para teste e simulação de sinais em equipamentos. Dentre as diversas aplicações de seus produtos, encontram-se aquelas capazes de realizar a emulação de sensores como RTDs (*Resistance Temperature Detector,* ou, em português, detector de termorresistência) e Strain Gauges. Estes módulos trabalham com múltiplos canais de resistências e operam em uma extensa faixa de valores programáveis pelo usuário. (PICKERING TEST INTERFACES), (PICKERING TEST INTERFACES, 2019).

Basicamente, as placas são compostas por séries de resistores que são incluídos ou excluídos da rede através da comutação de relés, como apresentado na Figura 7.

Figura 7 – Cadeia de resistores



Fonte: (OTTE, 2014)

Assim, é possível encadear estas redes com diferentes valores em uma placa. A combinação das várias cadeias viabiliza a geração de quaisquer valores de resistência. (OTTE, 2014).

Como exemplo comercial de uma aplicação desse tipo, cita-se a placa 40-251-044 da Pickering Interfaces, que possui dois canais de resistência configuráveis em uma faixa que varia 2,5 Ω a 1,51 MΩ com uma resolução de 2 Ω. Ademais, conta com a possibilidade de simulações de curto-circuito e circuito aberto em seus canais. Tal placa tem o valor estimado de US$3922,00, podendo ser inserida nas mais diversas aplicações. (PICKERING INTERFACES, 2020).

2.4.2 Simulador de termistores NTC Wireflow

O módulo de resistência programável WF314 fabricado pela Wireflow é um equipamento que permite a emulação de quaisquer sensores resistivos, como termistores NTC (*Negative Temperature Coefficient*, ou, em português, Coeficiente Negativo de Temperatura), PTC (*Positive Coefficient Temperature,* ou, em português, Coeficiente Positivo de Temperatura), ou, ainda os RTD já mencionados na seção anterior. O módulo permite a comunicação com uma aplicação desenvolvida em Labview, na qual pode ser inserida a temperatura desejada e o cálculo da resistência equivalente efetua-se segundo alguns parâmetros de entrada.

2.4.3 Multímetros Digitais

Equipamentos comumente conhecidos por sua versatilidade, os multímetros digitais ou DMMs (*Digital Multimeters,* ou, em português, Multímetros Digitais) são dispositivos de medição de grandezas elétricas. Atualmente, existem diversas modalidades destes equipamentos, que se diferem quanto as funções disponíveis e resolução, entre outros parâmetros, como se observa nos diferentes modelos na Figura 8. De modo geral, esse tipo de equipamento utiliza conversores A/D (Analógico Digital) para realizar a medição. Estes conversores, por sua vez, utilizam uma tensão de referência e retornam uma saída com base na comparação realizada. (LEI-GAO, CHUN-PING e MING, 2010).

Figura 8 – Multímetros Digitais

Fonte: (KEYSIGHT TECHNOLOGIES, 2020)

Ao receber a grandeza a ser medida, o valor é convertido para uma tensão específica dentro da faixa de operação do conversor A/D. Após, ao passar pelo conversor, é realizada a leitura e, assim, a interpretação para obtenção do valor medido. (LEI-GAO, CHUN-PING e MING, 2010)

2.4.4 Módulos de entradas e saídas

Os módulos de entradas e saídas são dispositivos que possuem a capacidade de realizar leituras de tensão ou aplicar saídas de tensão. Existem inúmeras opções de módulos disponíveis e que diversificam-se na topologia, aplicação e forma de operação. Para as saídas, existem modalidades não isoladas, suscetíveis à ruídos e com aplicação em dispositivos TTL (*transistor-transistor logic,* do português, lógica transistor-transistor) ou simplesmente circuitos que operem com 5 V ou 3.3 V. Outras opções incluem circuitos isolados galvanicamente através de opto-acopladores, topologias que utilizam relés, lógicas de acionamento, e demais aplicações variadas. Os módulos de entradas seguem a mesma orientação, apresentando uma vasta linha de opções disponíveis. Um módulo comercial é apresentado na Figura 9. (CONTEC COMPANY, 2020).

Figura 9 – Módulo E/S

Fonte: (IMPAC INSTRUMENTOS DE MEDIÇÃO, 2020)

2.5 Microcontroladores

A escolha do modelo a ser utilizado é fundamental, visto que sob uma perspectiva geral, os microcontroladores do presente trabalho são responsáveis por aquisição de dados, tratamento e intercomunicação via protocolo de comunicação serial.

Microcontroladores são dispositivos que integram hardware e software, podendo ser controlados via linguagem de programação específica. Ou seja, é possível, através de linguagens como C, realizar a implementação de diversas funções de controle em um hardware específico. (SOUZA, 2009).

A família PIC (*Programmable Interface Controller,* ou, em português, Controlador de Interface Programável) de microcontroladores, pertencentes a Microchip, iniciaram sua jornada no Brasil na década de 90 utilizando a empresa Artimar como meio de distribuição. Tais processadores, semelhantes aos microcontroladores ARM (*Advanced RISC Machine,* ou, em português, Máquina RISC Avançada) são máquinas RISC (*Reduced Instruction Set Computer,* ou, em português, Computador com um conjunto reduzido de instruções) e empregam a arquitetura Harvard, que possui barramentos individuais, para execução de instruções e para acesso de dados. (SOUZA, 2009).

No caso do ARM, a primeira versão disponibilizada ao mercado foi a ARM2, em 1987. Entretanto, a história destes microcontroladores se originou anos antes, devido a necessidade apresentada na época em termos de latência para execução de instruções, e custo para o desenvolvimento de projetos da empresa Acorn Computer Group. Por essa razão, após alguns, a própria Acorn promoveu uma arquitetura de processadores de 32 bits, dando o marco inicial na história desta linha de processadores. (RYZHYK, 2006).

Devido ao baixo consumo e simplicidade, a ARM é uma das arquiteturas mais utilizadas pelas maiores fabricantes de microcontroladores do mundo, a exemplo da STMicroelectronics, NXP e Texas Instruments, estando presente na maioria dos dispositivos portáteis comercializados atualmente. (ARM LIMITED, 2005), (PEREIRA, 2007).

Dessa forma, os microcontroladores ARM são máquinas RISC, tratando-se de CPUs de 32 bits, caracterizados principalmente pela velocidade e baixo custo, além de apresentar fácil migração de uma arquitetura para outra. Até a versão 7, as

máquinas faziam uso, essencialmente, da arquitetura Von Neumann, que compartilha um barramento apenas. Atualmente, já se dispõe de versões (8-A por exemplo) que empregam o uso da Arquitetura Harvard. (PEREIRA, 2007), (ARM LIMITED, 2005).

2.6 Kits de prototipagem e desenvolvimento

2.6.1 Arduíno

O Arduíno é uma plataforma *open source* (ou, em português, fonte aberta) para prototipagem de hardware e software que se popularizou, principalmente, pela facilidade de utilização, simples linguagem de programação (diferente dos diversos microcontroladores disponíveis no mercado) e recursos que apresenta. (ARDUÍNO, 2019).

Surgiu em 2005 mediante pesquisas realizadas por estudantes do Instituto Ivrea, na Itália, que tinham como principal objetivo uma opção *open source* para profissionais da área e entusiastas, de modo a compatibilizar diferentes tipos de sensores e atuadores. Hoje, a plataforma conta com inúmeras placas de desenvolvimento para as mais diversas aplicações. (NAYYAR e PURI, 2016).

2.6.2 Nucleo STMicroeletronics

Similarmente ao Arduino, a STMicroelectronics dispõe de diversos modelos de placas de desenvolvimento, que variam recursos e entre si. Em geral, essas placas são mais poderosas do que as placas do Arduino e possuem menor custo, porém, necessitam de um maior estudo para utilização. (STAFF, 2016). A Tabela 1 apresenta uma comparação entre modelos de Arduino e núcleos STMicroeletronics.

Tabela 1 – Comparação Arduino x Nucleo

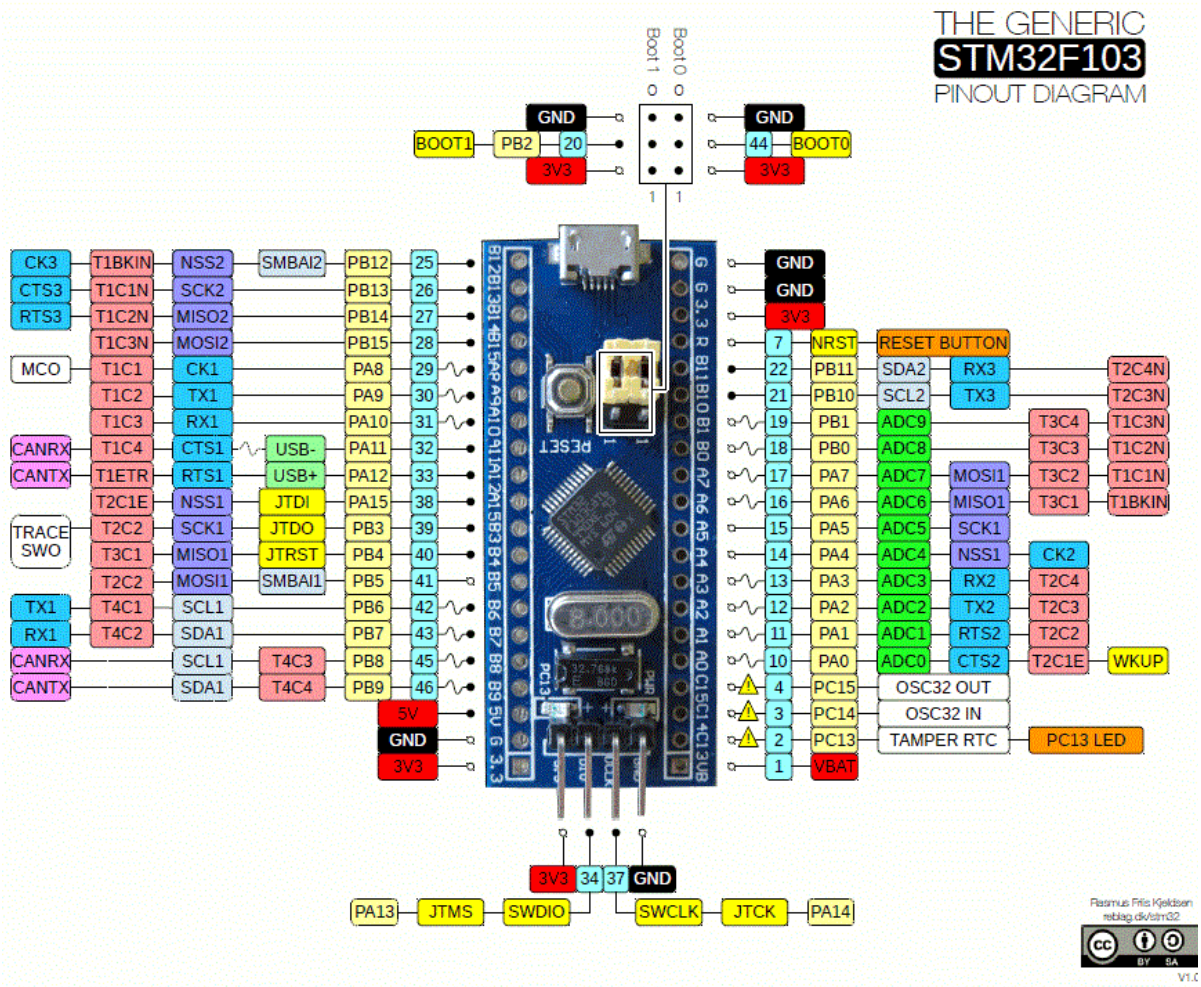
	NUCLEO F401	Arduino UNO	NUCLEO F303	Arduíno Nano
Microcontrolador	STM32F401 32-bit	ATMEGA 328 8-bit	STMF303K8 32-bit	ATMega 328 8-bit
Família	ARM Cortex M4	AVR	ARM Cortex M4	AVR
Frequência do Clock	84 MHz	16 MHz	72 MHz	16 MHz
Memória Flash	521 kb	32 kb	64 kb	32 kb
SRAM	96 kb	2 kb	16 kb	2 kb
Memória EEPROM	-	1 kb	N/A	1 kb
PWM	10	6	6	6
Entradas analógicas	16	6	10	8
Pinos digitais	47	14	13	14
Módulos I2C	3	1	1	1
Módulos USART	3	1	2	1
Módulos SPI	4	1	1	1
Timer	10	3	5	3
Unidade de ponto flutuante	Um	N/A	Um	N/A
Tensão (máxima)	5 V	5 V	5 V	5 V
USB OTG	Um	N/A	N/A	N/A
Dimensões	68x80mm	53x68mm		45x18mm
Preço ($)	14	25	10	15

Fonte: (STAFF, 2016)

É possível verificar que o custo do Arduino é mais alto em relação aos núcleos STM32, e estes, por sua vez, possuem uma performance muito superior em termos de hardware e ferramentas disponíveis.

Além das placas NUCLEO STM32, existem outras com diferentes funcionalidades, e a escolha depende da necessidade do desenvolvedor. A placa de desenvolvimento que utiliza o microcontrolador STM32F103C8T6, um ARM 32-bit Cortex M-3, é denominada *bluepill,* cuja identificação dos pinos é apresentada na Figura 10. Essa placa dispõe dos requisitos mínimos para programação e é compatível com todas as IDEs (*Integrated Development Environment,* ou, em português, ambiente de desenvolvimento integrado), comumente utilizadas para microcontroladores STM32, além de também ser compatível com a IDE do Arduino. (ASWINTH, 2018). A Figura 10 mostra a distribuição dos pinos desta placa.

Figura 10 – Placa de desenvolvimento STM32F103C8



Fonte: (ASWINTH, 2018)

2.7 Protocolo de comunicação e detecção de erros

2.7.1 Protocolo serial

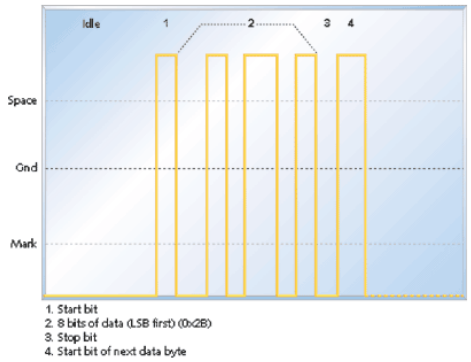
O protocolo serial é a mais simples e utilizada interface de comunicação, estando presente em uma grande gama de equipamentos, o que facilita sua implementação em sistemas. (EMBEDDED STAFF, 2002).

O modo de operação deste protocolo pode ser realizado na forma *full-duplex*, que permite dados que podem ser recebidos e transmitidos por meio de duas linhas de comunicação distintas para cada função. Outra forma, a *half-duplex*, aplica uma mesma linha para transmissão e recepção de dados, não permitindo, portanto, que as ações sejam executadas simultaneamente. Existem outras maneiras, mas estas não

são tão utilizadas no âmbito de desenvolvimento. Ademais, é possível configurar para a comunicação serial para o modo síncrono ou assíncrono. O modo síncrono sincroniza a transmissão e recepção de dados com o *clock*, enquanto o assíncrono não. (EMBEDDED STAFF, 2002).

O protocolo apresenta diversos modos de interfaceamento e configuração, dentre estes o TIA/EIA-232-F, conhecido como RS-232, que trata a forma mais popular e presente em computadores, podendo atingir velocidades até 115,2 kbps (*kilo bits per second*, ou, do português, quilo bits por segundo). (EMBEDDED STAFF, 2002). A Figura 11 apresenta uma típica comunicação RS-232, contemplando os bits de dados.

Figura 11 – Interface Serial RS-232



Fonte: (EMBEDDED STAFF, 2002)

2.7.2 Protocolo I2C

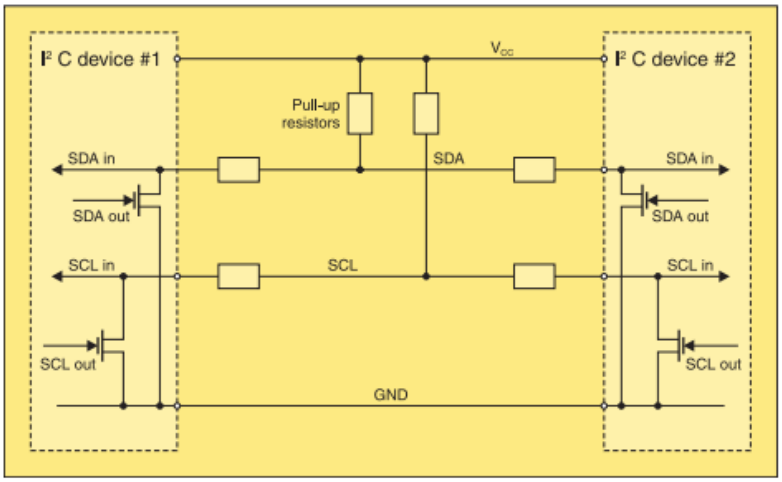
Surgindo na década de 80, pela Philips Semiconductors, o protocolo I2C (*Inter Integrated Circuit,* ou em português, Circuitos Interintegrados) ou simplesmente I2C, como é comumente conhecido, é uma interface serial de simples implementação que possui, frente à maioria dos protocolos universalmente utilizados, o endereçamento de dispositivos como principal característica, eliminando problemas que podem ser causados pelo compartilhamento de memória. Inicialmente, possuía o objetivo de interligar componentes de uma televisão a uma CPU (*Central Process Unit,* ou, em português, Unidade de Processamento Central). Hoje, grande parte dos circuitos integrados oferecem este recurso, sendo amplamente difundido no mundo da programação. Além da enorme vantagem, o I2C ainda apresenta outros benefícios

para sua utilização, tais como o baixo custo, boa performance, fácil migração, entre outros. (MYERS, 2007), (MANKAR, DARODE, *et al.*, 2014).

Basicamente, a implementação mais simples de um I2C é constituída por uma arquitetura mestre-escravo. Os dispositivos são conectados entre si por duas linhas: SDA (*Serial Data*, ou, em português, Data Serial) e SCL (*Serial Clock*, ou, em português, *Clock* Serial), além da linha conectada ao GND (*Ground,* ou, em português, Terra) dos dispositivos. A linha SDA é responsável por enviar dados para os dispositivos escravos ou receber dados destes, sendo do tipo *half-duplex*. Isto é, atua de forma bidirecional, porém não simultânea (mestre e escravo devem atuar distintamente). Já a linha SCL tem o papel de gerar o *clock*, que realiza a sincronia entre os dispositivos. (MANKAR, DARODE, *et al.*, 2014), (MYERS, 2007), (Myers, 2007), (LEENS, 2009).

Fisicamente, essas linhas normalmente possuem uma configuração de coletor-aberto. Em outras palavras, são conectadas internamente a base de um transistor NPN de forma que, ao serem acionadas, fazem o transistor conduzir, levando *GND* ao pino de saída. Em contrapartida, se a saída não estiver acionada, o pino de saída permanece aberto. Por conta disso, são necessários resistores de *pull-up* (ligados ao VCC) para levar o sinal a nível alto quando este se encontra aberto pela interface de coletor aberto (VALDEZ e BECKER, 2015). A Figura 12 abaixo apresenta uma típica interface I2C.

Figura 12 – Interface I2C

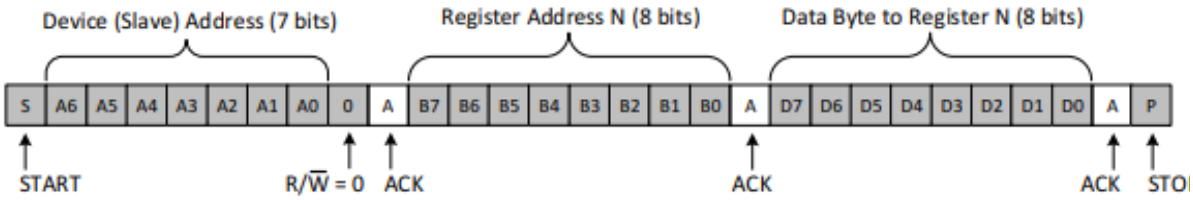


Fonte: (LEENS, 2009)

Sob uma perspectiva ampla, o protocolo atua da seguinte forma: o mestre indica que uma transmissão irá ocorrer através de um bit de *Start* para que os escravos estejam prontos para uma nova troca de informações. Logo depois, o mestre envia o endereço do dispositivo no qual deseja acesso juntamente a um bit de R/W (*read/write,* ou, em português, leitura ou escrita), a fim de definir se este escravo receberá algum dado do mestre ou apenas será lido por ele. Após, os escravos comparam o endereço recebido pelo mestre ao seu próprio endereço; se esta comparação resultar em alguma correspondência, o respectivo escravo gerará um sinal de reconhecimento, denominado *Acknowledge*, enviando-o ao mestre.

A partir disso, o mestre enviará o registrador em que se deseja enviar ou receber dados, de acordo com o a configuração ajustada em R/W. O escravo enviará um novo sinal de *Acknowledge*, indicando que está pronto para receber dados. Assim, o mestre passa a escrever ou ler dados do registrador selecionado e do endereço desejado. Uma vez que a transmissão foi executada, o mestre envia um bit de *Stop*, designando que o barramento de comunicação foi liberado. Os demais escravos, por não terem sido acessados, apenas aguardam o fim da transmissão. (MANKAR, DARODE, *et al.*, 2014), (MYERS, 2007), (Myers, 2007), (LEENS, 2009), (VALDEZ e BECKER, 2015). Um exemplo de topologia da comunicação apresentando o número de bits para cada etapa é apresentado na Figura 13.

Figura 13 – Escrita Barramento I2C



Fonte: (VALDEZ e BECKER, 2015)

O I2C, dentre as várias vantagens apresentadas, possui uma única limitação de número de dispositivos pela capacitância das linhas SDA e SCL. Em um modo típico de operação, é sugerido que a capacitância possua um máximo de 400pF por barramento. O valor desta capacitância normalmente é exposto nos documentos dos dispositivos I2C e influencia diretamente no cálculo do resistor de *pull-up* necessário para a comunicação e a velocidade máxima da mesma, que pode chegar até 3,4mb/s. (ARORA, 2015), (LEENS, 2009).

2.7.3 CRC

As linhas de transmissão, onde a troca de informações é realizada, são submetidas a condições que acabam por prejudicar a veracidade dos dados. Isso se deve, muitas vezes, as estruturas físicas da rede, velocidade de transmissão e ruídos externos, entre outros fatores que possam impactar na estrutura. (MATLOFF, 2001).

Para promover um uma rede de comunicação imune a erros deste tipo, existem técnicas de CRC (*Cyclic Redundancy Check*, ou, do português, Verificação Cíclica de Redundância), que têm a função de detectar de erros em dados digitais. Basicamente, esta técnica consiste em atrelar uma parcela de dados ao dado original total e transmitir juntamente. A parcela é resultado de um cálculo realizado com base nas demais informações desse mesmo dado. Ao enviar o dado, a interface de recepção verifica as informações recebidas, calcula o CRC recebido de forma similar a anterior e compara com a parcela recebida. (MATLOFF, 2001), (TECHOPEDIA, 2016).

2.8 Aquisição de dados

Dispositivos de aquisição de dados, ou do inglês, DAQ (*Data Acquisition*) consistem na realização da medição de uma grandeza física, interpretação do sinal lido e na transformação do mesmo em um padrão pré-definido para envio da informação obtida a uma IHM. O processo é mostrado na Figura 14. (NATIONAL INSTRUMENTS, 2019).

Figura 14 – Diagrama de blocos de uma DAQ

Fonte: (NATIONAL INSTRUMENTS, 2019)

Estes sistemas são utilizados nas mais variadas aplicações, como no controle de processos, monitoramento contínuo e testes de produção. Todas as informações necessárias são captadas por sensores, que têm a função básica de converter uma

grandeza física para uma grandeza elétrica (transdutores de pressão, sensores de temperatura, acelerômetros, entre outros). (NATIONAL INSTRUMENTS, 2019).

Os sinais obtidos passam por um condicionador, isto é, são modificados para que a leitura do sinal seja efetuada apropriadamente. Visto que muitos dos sensores utilizados possuem comportamento não linear, um condicionador de sinais pode realizar a linearização, de maneira a deixar a medição mais precisa. Outra etapa importante, também realizada por um condicionador, trata da readequação do sinal obtido para deixá-lo apropriado à faixa de utilização do conversor (etapa posterior). Esta readequação pode ser efetuada através de um bloco de amplificação ou atenuação, de acordo com o sensor utilizado e do conversor.

Assim, é aplicado um filtro a informação, para que somente dados válidos sejam considerados. Passando o condicionador de sinal, os dados são convertidos de um sinal contínuo para um digital. Todas estas tarefas são possíveis a partir do hardware e de um software que o controle. Este software é denominado software de driver, e tem a função específica de controlar o hardware de aquisição de dados, de modo a garantir a correta operação e comunicação entre o sistema de aquisição e a IHM, bem como os dados a serem enviados. (HBM, 2019), (NATIONAL INSTRUMENTS, 2019).

Em seguida, o sinal é enviado para uma IHM, que pode ser um computador, e os dados são apresentados de acordo com o software de aplicação desenvolvido. O software de aplicação, por sua vez, é a interface que interage com o usuário, viabilizando a visualização e entrada de dados, e a apresentação de gráficos entre as múltiplas funcionalidades desejadas. Esta pode ser desenvolvida com softwares específicos, como o Labview, da National Instruments. (NATIONAL INSTRUMENTS, 2019).

2.9 Técnicas para aquisição de dados

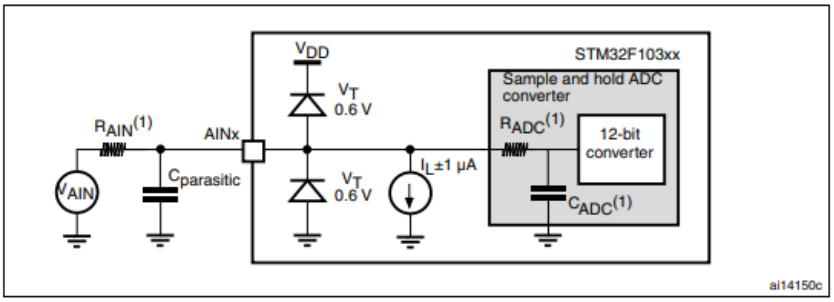
2.9.1 Medição de tensão

Na contemporaneidade, microcontroladores dispõem, em sua maioria, de conversores A/D (Analógico/Digital) integrados. De modo geral, estes conversores internos atendem as necessidades do usuário. Em caso de medições de alta precisão

ou funcionalidades muito específicas, podem ser utilizados circuitos integrados dedicados. (BILL, 2018).

Os conversores A/D normalmente operam conforme a Figura 9 abaixo. O capacitor interno C_{ADC} é o componente que define o tempo de amostragem de leitura de tensão entre os diversos canais de A/D do microcontrolador; o diodo superior conectado evita picos de tensão positivos, suavizando a curva superior do degrau de tensão; e o diodo inferior tem a função de evitar os picos negativos de tensão. O conjunto de diodos são denominados diodos de grampeamento e tem-se a exibição do circuito de proteção completo abaixo, na Figura 15. (DIGI-KEY ELECTRONICS, 2012).

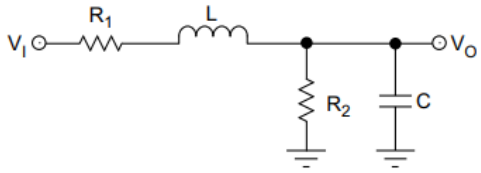
Figura 15 – Conexão típica de um conversor A/D



Fonte: (STMICROELETRONICS, 2015)

Usualmente, junto aos conversores A/D, são utilizados filtros para eliminação de ruídos e interferências. Estes filtros normalmente são constituídos de circuitos RLC (Resistores, Capacitores e Indutores) para formar a configuração passa-baixas (STEFFES, 2005), assim como mostrado na Figura 16.

Figura 16 – Filtro RLC

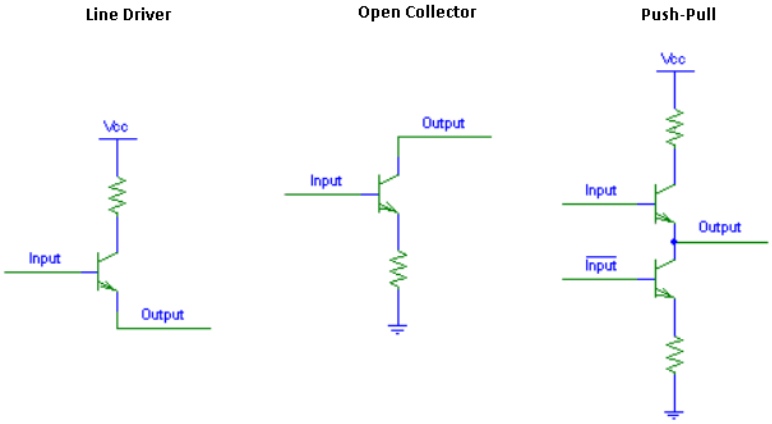


Fonte: (STEFFES, 2005)

2.9.2 Topologias de Saídas

Assim como as entradas, circuitos de saída apresentam variadas topologias para diferentes aplicações, sendo que algumas das mais populares são apresentadas na Figura 17. *Line driver* (driver de linha) se refere a um tipo de saída na qual, quando ativa, possui *VCC* (voltage common collector, ou, do português, tensão de coletor comum) na saída e, quando inativa, mantém a saída "flutuando". A saída de *open collector* (coletor aberto) utiliza o mesmo conceito, porém, quando ativa, mantém *GND* em sua saída. Por fim, a saída de *push-pull* (empurra-puxa) sempre possui algum estado determinado, isto é, se a entrada estiver com nível alto, o transistor superior irá conduzir, levando *VCC* a saída, enquanto o transistor inferior permanece desligado. Por outro lado, caso a entrada esteja com nível lógico baixo, a entrada negada no transistor inferior irá conduzi-lo, fazendo com que o transistor superior desligue e GND seja levado a saída. (NATIONAL INSTRUMENTS, 2019).

Figura 17 – Topologias de saídas



Fonte: (NATIONAL INSTRUMENTS, 2019)

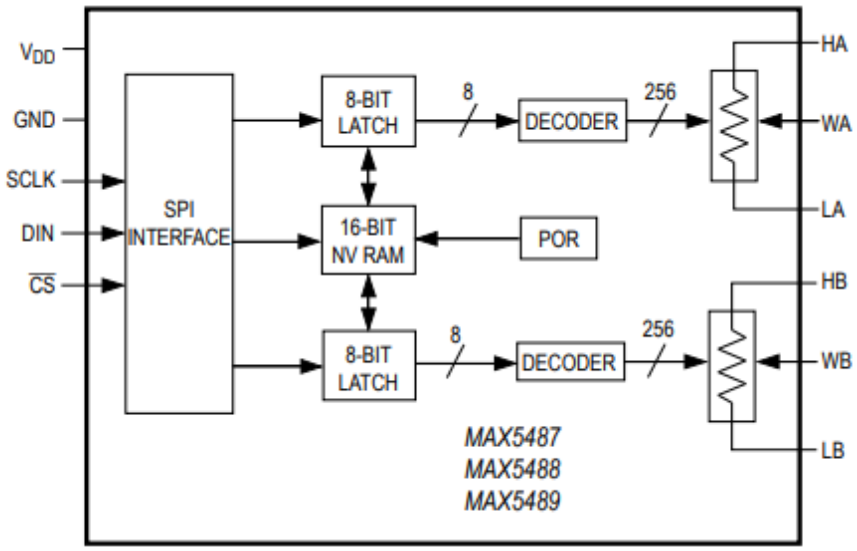
2.9.3 Potenciômetro digital

Como alternativa aos módulos de resistência programável típicos, que, conforme visto anteriormente, são controlados via chaveamento de resistências – possuindo elementos de chaveamento e precisão de alto custo, temos os dispositivos conhecidos como potenciômetros digitais. Estes componentes são controlados via barramento serial a fim de determinar uma resistência de saída. (ON SEMICONDUCTOR, 2013).

Internamente, o componente é estruturado com uma interface serial, que realiza a leitura e a envia para a circuito *latch* (circuito sequencial), que guarda a informação. Em seguida, o dado é enviado a decodificador/multiplexador com o propósito de ajustar a resistência selecionada. Existem diversos modelos, com diferentes valores nominais de resistência e resoluções disponíveis.

Abaixo, na Figura 18, é apresentado um exemplo de potenciômetro digital MAX5487/MAX5488/MAX5489, que utiliza interface SPI para controle. (ON SEMICONDUCTOR, 2013).

Figura 18 – Diagrama de funcionamento de potenciômetro digital



Fonte: (MAXIM INTEGRATED PRODUCTS, 2010)

2.10 Microcomputadores

Esta plataforma designa-se por um computador que possui recursos heterogêneos e baixo custo. Diferente dos microcontroladores atuais, visto a Figura 19, possibilita conexão com internet, saídas de vídeo, comunicação USB (*Universal Serial Bus,* ou, em português, Porta Universal), I2C e SPI, pinos de entradas e saídas.

Figura 19 – Raspberry Pi 4 Model B

Fonte: (RASPBERRY PI ORG, 2019)

Deste modo, trata-se de um computador com flexibilidade de utilização: seja para aplicações de amostragem de dados, para IHM, entre outras possibilidades. (RICHARDSON e WALLACE, 2013).

3 METODOLOGIA

Os itens a seguir são orientados ao desenvolvimento do projeto em questão, envolvendo os materiais e métodos necessários para a sua implementação, códigos desenvolvidos pertinentes a aplicação, técnicas necessárias, topologias e componentes adotados, construção do protótipo e a IHM, considerando os conteúdos referenciados anteriormente.

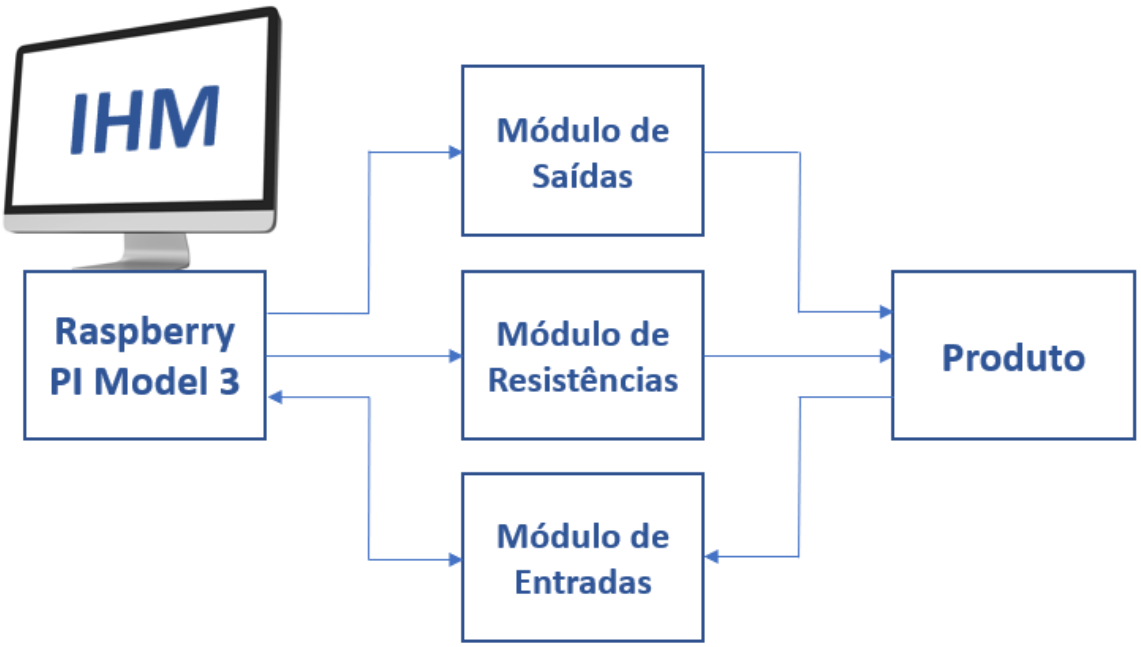
Na Figura 20 é apresentado o diagrama de blocos geral do protótipo, destacando-se que a direção das setas é fundamental para a compreensão do sistema. A IHM possibilita o controle e a exibição dos dados de teste, utilizando um dispositivo de hardware dedicado que é interligado com os demais módulos.

O módulo de saídas recebe a informação da IHM indicando a quantidade e tipos de saídas necessárias, isto é, ele interpreta e cria estas saídas, que servirão como entradas para um determinado produto. De forma semelhante, o módulo de sensores de temperatura executa esta tarefa, mas criando canais de resistência configurável que possuem a função de emular as entradas de sensores de temperatura do controlador. A IHM é, portanto, capaz de operar estes dois módulos.

O módulo de entradas, por sua vez, recebe o comando da IHM também com a informação de quantidade e os tipos de saídas do produto. Esse comando da IHM é interpretado e, em seguida, o módulo habilita a leitura das saídas correspondentes do produto. Posteriormente, retorna estes dados de leitura para a própria IHM.

A rede é toda estruturada via protocolo serial RS-232 com detecção de erros CRC, a fim de padronizar componentes e a rede em geral.

Figura 20 – Diagrama de blocos de funcionamento do protótipo.



Fonte: Elaborado pelo autor.

3.1 Definições de hardware e software

O software escolhido para a IHM foi o ambiente QT Creator na plataforma Raspberry Pi Model 3 utilizando o sistema Raspberry Pi OS que, além de apresentar uma ampla biblioteca de materiais e tutoriais disponíveis para desenvolvimento, ainda é um conjunto totalmente portátil, oferecendo acesso direto aos pinos de E/S, comunicação, entradas USB e conexão HDMI. Para alimentação do conjunto, se tratando de um protótipo, foi utilizada uma fonte ATX (*Advanced Technology eXtended*, ou, do português, tecnologia avançada estendida) comum modificada, disponibilizando as tensões de 3.3 V, 5 V e 12 V, conforme apresentado na Figura 21.

Figura 21 – Fonte de alimentação do protótipo

Fonte: Elaborado pelo autor.

Os módulos, por sua vez, empregam a placa de desenvolvimento *bluepill*, que possui um microcontrolador STM32F103C8T6 da STMicroeletronics, tendo disponível protocolos de comunicações I2C, SPI, serial ou mesmo CAN (este último interessante para aplicações automotivas futuras), conversores A/D, *timers* e outros periféricos, utilizando o ambiente de desenvolvimento STM32CubeIDE, que une uma IDE de desenvolvimento ao STM32CubeMX, facilitando a configuração dos pinos do microcontrolador. Para algumas aplicações específicas, seria conveniente a seleção de um microcontrolador com características propícias para a determinada função, como, por exemplo, o módulo de sensores de temperatura: foi necessário apenas a rede I2C e pinos de E/S para a confecção deste dispositivo, o que significa que um microcontrolador muito mais simples poderia ter sido utilizado. Entretanto, tratando-se de um protótipo, este microcontrolador atende os requisitos iniciais.

Os demais componentes e topologias adotadas são detalhadas na sua respectiva seção.

3.2 Desenvolvimento IHM

A IHM foi estruturada de forma a emular um sistema automotivo cujos dados possam ser visualizados em tempo real. É possível interagir com o *DUT (Device Under Test*, ou, do português, dispositivo em teste) enviando comandos para a operação dos módulos e cadastro de novos produtos.

A primeira tela desenvolvida foi o menu principal, conforme mostrado na Figura 22, o qual oferece duas possibilidades ao usuário: iniciar um teste já existente ou cadastrar um novo.

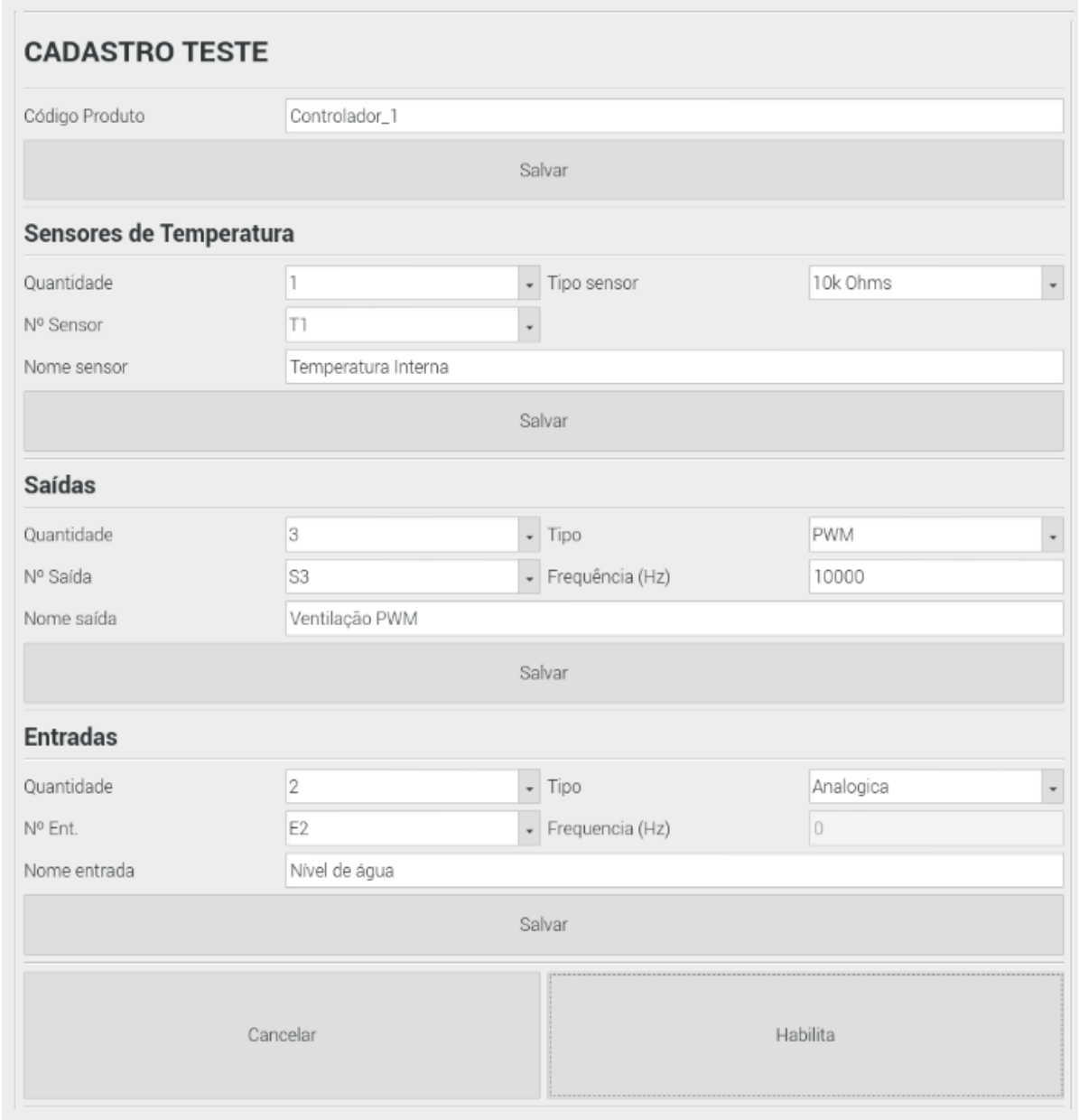
Figura 22 – Tela inicial

Fonte: Elaborado pelo autor.

Ao clicar em cadastro de novo teste, devem ser informadas, de acordo com a tela apresentada na Figura 23, as características do produto, fornecendo dados tais como código do produto, quantidade e tipos de saídas, quantidade e tipos de entrada, e quantidade e tipos de sensores de temperatura. Vale ressaltar que estas informações são referentes diretamente ao produto, por conseguinte, relativas as entradas e saídas do próprio controlador. Em geral, a tela é dividida em dois blocos: a tela de cadastro propriamente dita, apresentada na Figura 23, e a de pré-visualização do teste, mostrada na Figura 24. A tela completa é exibida no APÊNDICE A – Tela de cadastro de novo teste.

.

Figura 23 – Tela de cadastro de novo teste.

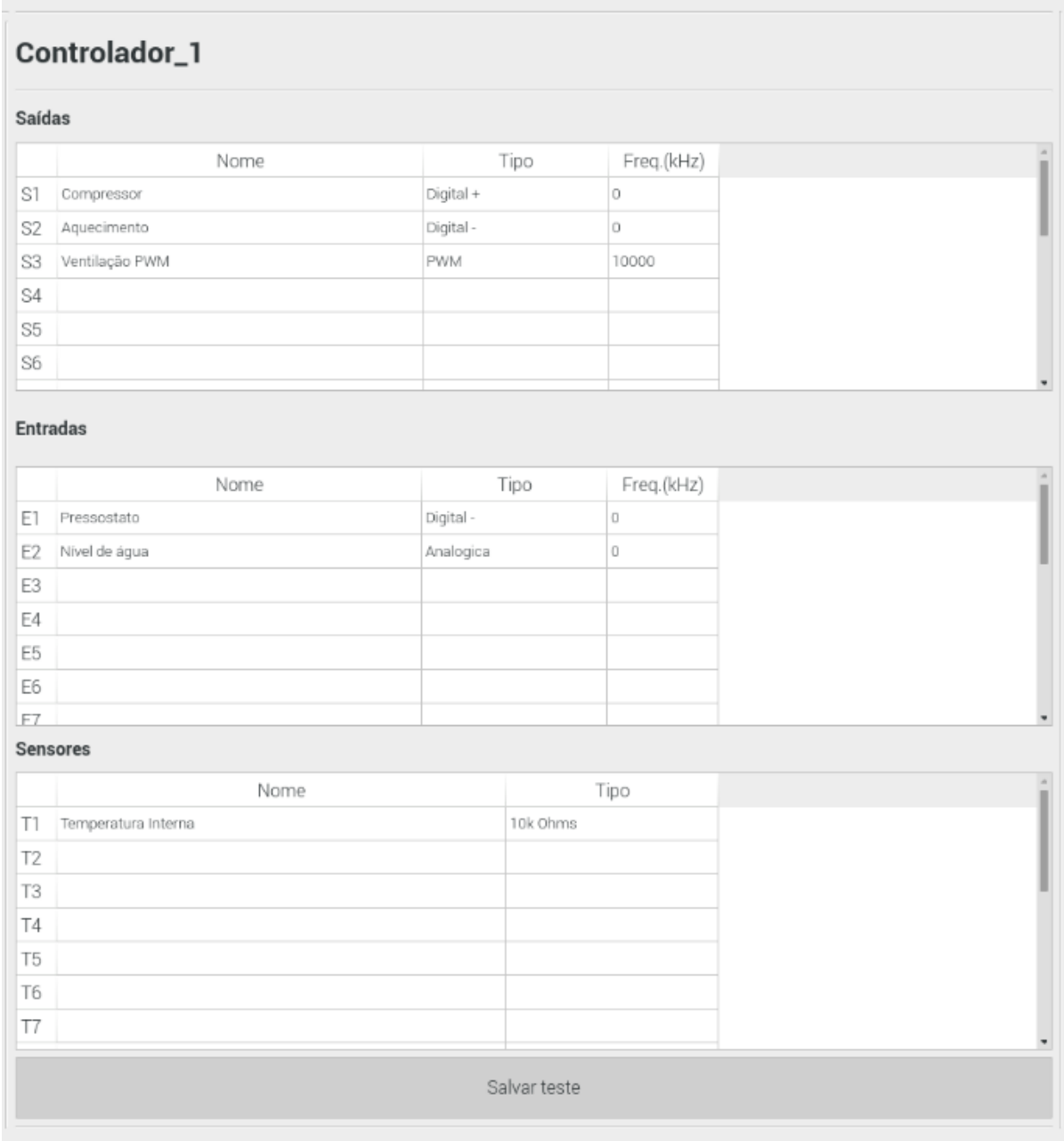


Fonte: Elaborado pelo autor.

A IHM possibilita a configuração de 16 sensores de temperaturas, 16 entradas e 16 saídas, nas quais os sensores são constituídos da indicação “T” e o número do sensor, as entradas de “E” e o número da entrada e as saídas de “S” juntamente ao número da saída. O campo “Tipo” em “Sensores de Temperatura” é meramente uma indicação do tipo de sensor em que se está trabalhando, enquanto os tipos de entrada e saída determinam qual configuração de leitura ou de saída deve ser realizada. Para os itens cuja configuração é PWM, o campo de “Frequência” é habilitado para preenchimento, não sendo possível manter o valor zero. Cada item pode ser salvo

separadamente através da tecla “Salvar”, e a tecla “Habilita” só é habilitada se, de fato, o código do produto ter sido salvo e mostra, conforme exibido na Figura 24, uma pré-visualização para conferência dos dados preenchidos. Se o usuário cometer algum erro de cadastro, basta selecionar o número da entrada, saída ou sensor desejado, preencher e habilitar novamente para a verificação dos novos dados.

Figura 24 – Tela de seleção de produto

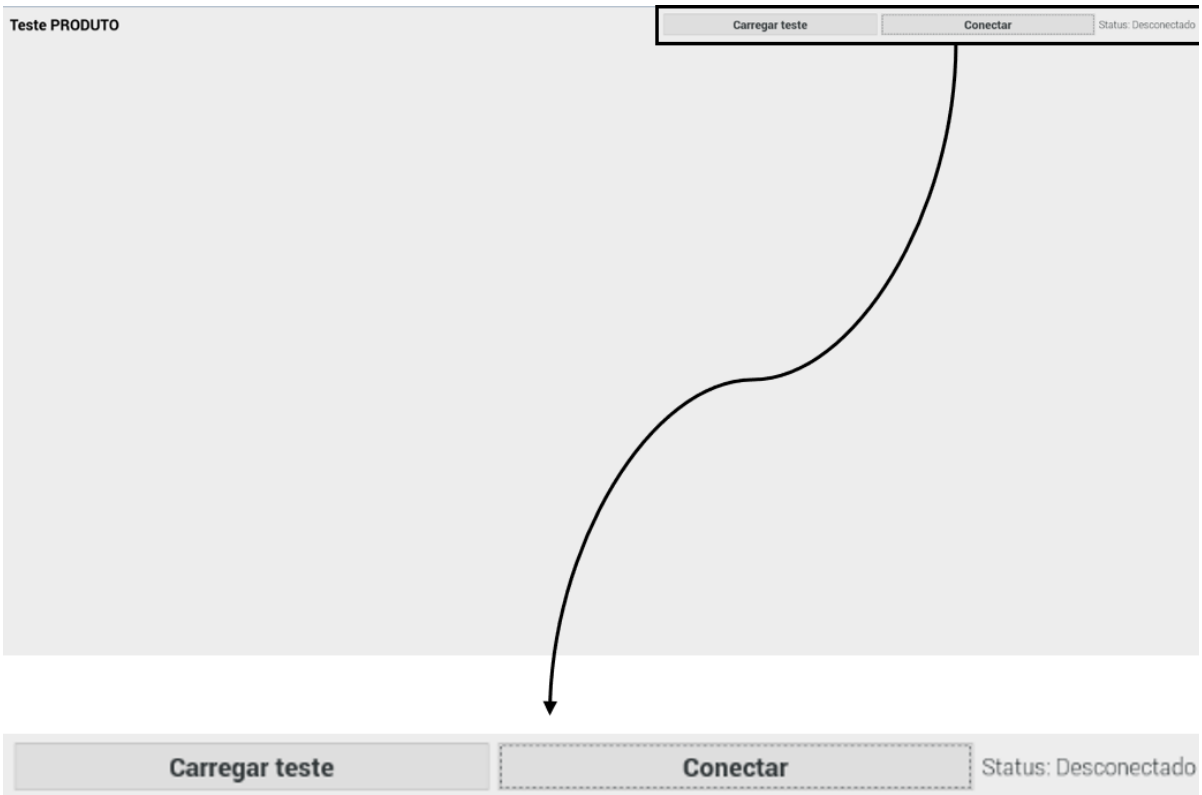


Fonte: Elaborado pelo autor.

Ao clicar na tecla “Salvar Teste”, os dados são armazenados ordenadamente em um arquivo .txt, que será lido na tela de teste. Cada linha do arquivo corresponde a exata posição de um determinado dado.

Ao salvar o teste e fechar a tela de cadastro, voltamos a Figura 22, onde é possível clicar na tecla “Iniciar teste”. A tela que surge em seguida é apresentada em branco com os botões de “Carregar teste” e “Conectar” habilitados e o “Status: Desconectado” referente a porta serial, conforme mostrado na Figura 25, indicando, então, que a IHM está aguardando carregamento de um arquivo e a conexão com a porta serial.

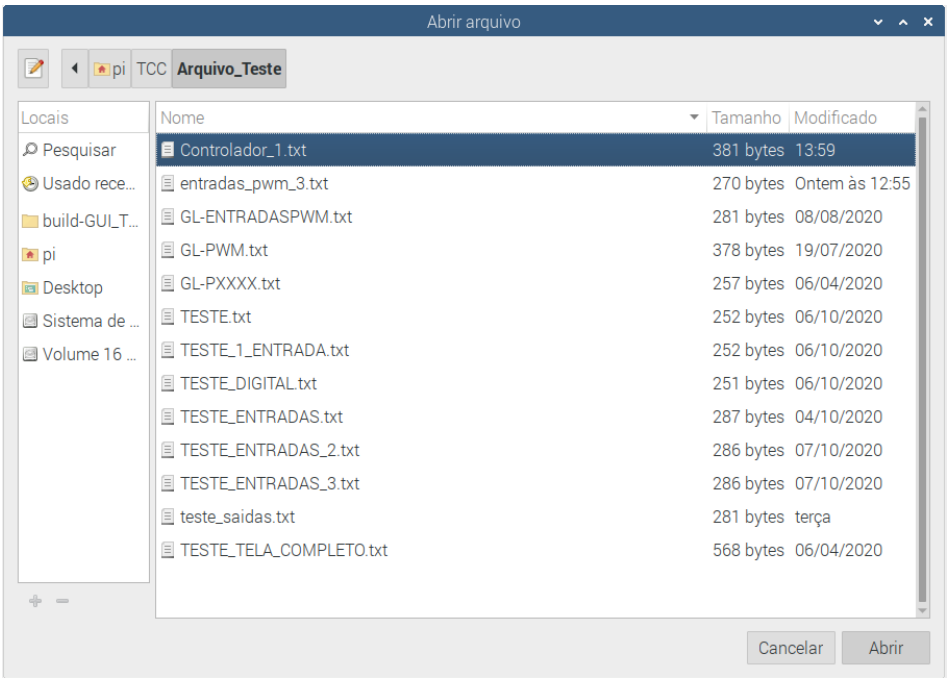
Figura 25 – Tela de teste de produto em branco.



Fonte: Elaborado pelo autor

Clicando-se em “Carregar teste”, vamos a tela de carregamento apresentada na Figura 26. Nela, estão contidos todos os arquivos de testes anteriores correspondentes a diferentes tipos de controladores, salvos em um diretório padrão específico selecionado para armazenamento.

Figura 26 – Tela de carregamento de arquivo.

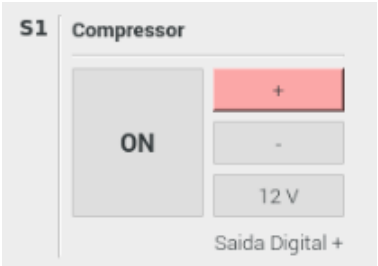


Fonte: Elaborado pelo autor

Selecionando o arquivo desejado e apertando em “Abrir”, o teste configurado será carregado. A IHM, em seguida, lê as posições do arquivo escolhido e se configura automaticamente de acordo com esta leitura, inserindo todas as informações das saídas, entradas e sensores (nesta ordem), no sentido de cima para baixo e esquerda para direita. A tela de teste completa carregada é apresentada no APÊNDICE B – Tela de teste carregada. A seguir, são apresentadas as configurações possíveis de entradas, saídas e sensores de temperaturas e suas respectivas informações.

a) Saídas digitais positivas: possuem a indicação vermelha no sinal “+”, a leitura de tensão e a indicação “*ON*”, “*OFF*” ou “*OPN*”, isto é, 12 V, 0 V ou circuito aberto, respectivamente, assim como apresentado na Figura 27.

Figura 27 – Exemplo saída digital positiva



Fonte: Elaborado pelo autor

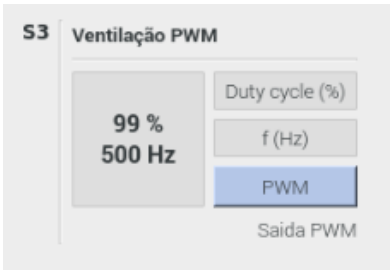
b) Saídas negativas: possuem uma indicação verde no sinal “-“, uma leitura de tensão e a indicação “*ON*”, “*OFF*” ou “*OPN*”, isto é, 0 V, 12 V ou circuito aberto, respectivamente. Observa-se que a relação “*ON*” e “*OFF*” é exatamente a oposta das saídas positivas, conforme a Figura 28.

Figura 28 – Exemplo saída digital negativa

Fonte: Elaborado pelo autor

c) Saídas PWM: possuem uma indicação azul com a denominação PWM, a indicação de *duty cycle* com o seu percentual e a em frequência em Hertz, de acordo com o exibido na Figura 29.

Figura 29 – Exemplo saída digital PWM



Fonte: Elaborado pelo autor

d) Saídas Analógicas: possuem uma indicação amarela com a denominação analógica juntamente a tensão lida, como é possível observar na Figura 30.

Figura 30 – Exemplo saída analógica

Fonte: Elaborado pelo autor

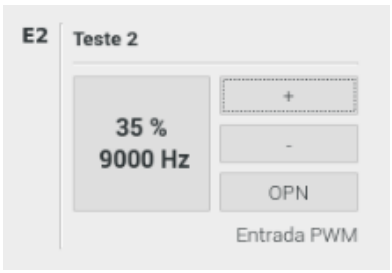
e) Entradas digitais positivas ou negativas: viabilizam o ajuste de três estados através dos botões: circuito aberto, VCC (12 V) ou GND (0 V), conforme mostrado na Figura 31.

Figura 31 – Exemplo entrada digital

Fonte: Elaborado pelo autor

f) Entradas PWM: possibilitam o ajuste do *duty cycle* de 0 % a 100 % ao passo de 5% sob uma frequência fixa, definida pelo arquivo carregado. Ainda é possível abrir o circuito através da tecla “OPN”, conforme a Figura 32.

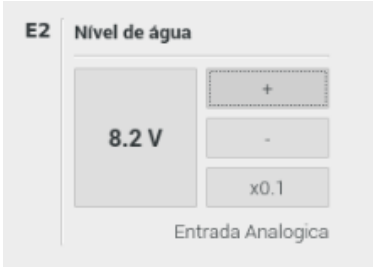
Figura 32 – Exemplo entrada PWM



Fonte: Elaborado pelo autor

g) Entradas analógicas: possibilitam o ajuste de uma tensão de 0 V a 12 V ao passo de 0,1 ou 1, conforme a tecla disponível, assim como exibido na Figura 33.

Figura 33 – Exemplo entrada analógica



Fonte: Elaborado pelo autor

h) Sensores de temperatura: possibilita o ajuste de uma resistência de 0 Ω a 999999 Ω ao passo de 1, 100, 10000 ou 100000 através da tecla indicada como “x1”, conforme mostrado na Figura 34.

Figura 34 – Exemplo sensor de temperatura

Fonte: Elaborado pelo autor

Para que a IHM inicie sua operação, basta clicar na tecla “Conectar” e assim a conexão com os módulos é efetuada e o controle iniciado.

3.3 Protocolo de comunicação

3.3.1 Configuração serial

Conforme citado anteriormente, o protocolo implementado foi o serial RS-232. A configuração adotada nos módulos foi padronizada: utilizando uma taxa de transmissão de 19200 kbps*, full-duplex*, isto é, utilizando duas linhas distintas (uma para transmissão e outra para recepção de dados), no modo assíncrono, não havendo necessidade de sincronização com o *clock*. Outro ponto interessante foi a utilização do DMA *(Direct Memory Access*, ou, do português, Acesso de memória direta)*,* que permite otimização do código por se tratar de um bloco que possui acesso direto à memória, sem a obrigação de intervenção da CPU. (STMICROELETRONICS, 2015)

Desta forma, a comunicação foi implementada utilizando uma interrupção por DMA no bloco USART (*Universal Synchronous Asynchronous Receiver Transmitter*, ou, do português, Transmissor/Receptor Universal Síncrono e Assíncrono). A Figura 35 mostra a implementação de transmissão e recepção de dados nos módulos. A definição “TAM_MSG” se refere a quantidade de dados que serão recebidos.

Figura 35 – Implementação serial módulos

```
HAL_UART_Receive_DMA(&huart1, dadoRecebido, TAM_MSG);
HAL_UART_Transmit_DMA(&huart1, dadoEnviado, TAM_MSG);
```

Fonte: Elaborado pelo autor.

Por parte da Raspberry Pi, utilizando o QT Creator, não foi necessária configuração prévia, bastando fazer uso de um comando para inicialização da porta serial correspondente e posteriormente a conexão na velocidade de 19200 kbps adotada. (RASPBERRY PI ORG, 2020). A Figura 36 exibe a implementação realizada no QT Creator.

Figura 36 – Conexão porta serial QT Creator

```
if (flagSerial == 1)
{
    fflush(stdout);
    serialPutchar(fd, 0);
    serialClose(fd);
    flagSerial = 0;
    ui->LabelConexao->setText("Status: Desconectado");
    ui->Conectar->setText("Conectar");
    return;
}

flagSerial = 1;


if((fd = serialOpen("/dev/ttyS0", 9600)) < 0)
{
    ui->LabelConexao->setText("Erro");
    return;
}

if(wiringPiSetup() == -1)
{
  ui->LabelConexao->setText("Erro");
  return;
}


fflush(stdout);

QMessageBox::information(this,"Aviso","Conectado com sucesso");


    ui->LabelConexao->setText("Status: Conectado");
    ui->Conectar->setText("Desconectar");
```

Fonte: Elaborado pelo autor.

A “flagSerial” é aplicada globalmente para indicação de que a porta serial está conectada. A biblioteca “wiringSerial.h” dispõe das funções seriais utilizadas. A função “serialOpen()”, assim como “wiringPiSetup()” estão relacionadas a abertura e configuração serial e retornam o valor de -1 em situação de erros, caso contrário, a porta serial é conectada corretamente. (GORDON PROJECTS, 2012).

Para realizar o envio de dados, basta usar os comandos “serialPuts()” para textos ou “serialPutchar()” para caracteres. O recebimento de dados é realizado somente caractere a caractere. (GORDON PROJECTS, 2012).

Por fim, não foi implementado um circuito de comunicação auxiliar, visto que tanto Raspberry quanto *bluepill* utilizam um nível de tensão de 3,3 V. (RASPBERRY PI ORG, 2020).

3.3.2 Posicionamento de dados seriais

Os dados seriais são recebidos de forma ordenada, porém, em posicionamento deslocado. Isto é, se por exemplo for enviado “abcd” via serial, podemos receber “cdab”, onde a letra “a”, que inicialmente possuía a posição 1, agora ocupa a posição 3. Tem-se, nisso, a indicação de um deslocamento da informação, o qual mantém a ordem de recebimento correto: em ambas as situações a letra “a” vem depois da letra “d”, a letra “b” depois da letra “a”, e assim por diante.

Defronte a esse problema, a mensagem serial, tanto para os módulos quanto para a IHM, foi constituída utilizando índices formados pelos caracteres “/ x #”, onde x indica o número correspondente do dado, conforme exibido na Figura 37. Cada informação enviada ou recebida possui o prefixo que indica sua posição correta na mensagem.

Também verifica-se na Figura 37 que o prefixo possui sua posição fixa na informação, estando presente para todas as mensagens trocadas entre IHM e módulos. Os dados dos caracteres foram obtidos através da tabela ASCII (FEUP, 2003), que informa a correspondência do número 47 ao símbolo “/”, 35 ao “#” e de 49 a 52 aos números de 1 a 4. À vista disso, temos que o prefixo 1 trata do número do canal e o tipo de informação recebida (saída, entrada ou sensor). O prefixo número 2 pode informar um valor de tensão (multiplicado por 10 para não trabalharmos com números decimais) ou *duty cycle*, o prefixo número 3 informa valores de resistência ou frequência e, por fim, o prefixo 4, que contém o CRC recebido.

Figura 37 – Mensagem serial

```
dadoEnviado[0]  = 47;            /* Caracter /  */
dadoEnviado[1]  = 49;            /* Caracter 1  */
dadoEnviado[2]  = 35;            /* Caracter #  */
dadoEnviado[3]  = dado1;         /* Index Saida */
dadoEnviado[4]  = dado2;         /* Index Saida */

dadoEnviado[5]  = 47;            /* Caracter / */
dadoEnviado[6]  = 50;            /* Caracter 2 */
dadoEnviado[7]  = 35;            /* Caracter # */
dadoEnviado[8]  = dado3 >> 8;    /* Dado principal: Tensao * 10 ou Duty Cycle */
dadoEnviado[9]  = dado3;         /* Dado principal: Tensao * 10 ou Duty Cycle */

dadoEnviado[10] = 47;           /* Caracter / */
dadoEnviado[11] = 51;           /* Caracter 3 */
dadoEnviado[12] = 35;           /* Caracter # */
dadoEnviado[13] = dado4 >> 16;  /* Dado auxiliar: Frequencia ou Resistencia */
dadoEnviado[14] = dado4 >> 8;   /* Dado auxiliar: Frequencia ou Resistencia */
dadoEnviado[15] = dado4;        /* Dado auxiliar: Frequencia ou Resistencia */

dadoEnviado[16] = 47;              /*  Caracter / */
dadoEnviado[17] = 52;              /*  Caracter 4 */
dadoEnviado[18] = 35;              /*  Caracter # */
dadoEnviado[19] = CRC_Dados >> 24; /*     CRC     */
dadoEnviado[20] = CRC_Dados >> 16; /*     CRC     */
dadoEnviado[21] = CRC_Dados >> 8;  /*     CRC     */
dadoEnviado[22] = CRC_Dados;       /*     CRC     */
```

Fonte: Elaborado pelo autor.

Como o dado é recebido deslocado, conforme informado anteriormente, foi desenvolvido um algoritmo capaz de reposicionar os dados corretamente com base nos prefixos da mensagem. A Figura 38 mostra a função que executa a tarefa de reposicionar a mensagem. O primeiro laço de repetição percorre todas as posições da mensagem em busca do prefixo “/ 1 #, que é, de fato, a primeira posição da mensagem. O algoritmo também foi pensado para, se em caso de um caractere estar na última posição da mensagem, os dois primeiros serem verificados juntamente. Se dois ocuparem a penúltima e última posição, o primeiro será verificado juntamente.

Uma vez identificado, a posição é armazenada na variável “ordena” e a redistribuição dos dados é iniciada. As mesmas proteções para os valores limites na mensagem são empregadas também nesta etapa.

Figura 38 – Redistribuição da mensagem serial

```
void OrdenaMsgRecebida(void)
{
    int ordena    = 0;

    for (int i = 0; i <= (TAM_MSG-1); i++)
    {
        if (dadoRecebido[i] == 47) /* Caracter / */
        {
            if(i == (TAM_MSG-2))
            {
                if (dadoRecebido[i+1] == 49) /* Caracter 1 */
                {
                    if (dadoRecebido[i-(TAM_MSG-2)] == 35) /* Caracter # */
                    {   ordena = i;
                    }
                }
            }
            if(i == (TAM_MSG-1))
            {
                if (dadoRecebido[i-(TAM_MSG-1)] == 49) /* Caracter 1 */
                {
                    if (dadoRecebido[i-(TAM_MSG-2)] == 35) /* Caracter # */
                    {   ordena = i;
                    }
                }
            }
            if(i < (TAM_MSG-2))
            {
                if (dadoRecebido[i+1] == 49) /* Caracter 1 */
                {
                    if (dadoRecebido[i+2] == 35) /* Caracter # */
                    {   ordena = i;
                    }
                }
            }
        }
     }

    for (int i = 0; i <= (TAM_MSG-1); i++)
    {
    if (ordena == 0)
    {dadoTratado[i] = dadoRecebido[i + ordena];}

    if ((ordena > 0) && ((ordena + i) <= (TAM_MSG-1)))
    {dadoTratado[i] = dadoRecebido[i + ordena];}

    if ((ordena > 0) && ((ordena + i) > (TAM_MSG-1)))
    {dadoTratado[i] = dadoRecebido[i + ordena - TAM_MSG];}
    }
}
```

Fonte: Elaborado pelo autor.

3.3.3 CRC

A IDE STM32CubeIDE oferece uma implementação de *CRC* para a *bluepill*, que facilita a realização desta verificação. Entretanto, como a ferramenta não está disponível para o QT Creator, optou-se por uma detecção de erros mais simples.

Foi utilizado como base, portanto, o *CRC* do sensor DHT11 para implementação. Este sensor de temperatura e umidade envia uma soma de seus dados anexado a mensagem, que é verificada posteriormente (D-ROBOTICS UK, 2010). Assim, primeiramente o CRC recebido na serial é tratado e armazenado em uma variável, conforme exibido na Figura 39.

Figura 39 – CRC recebido

```
uint32_t CRC_Serial = (dadoTratado[(TAM_MSG-4)] << 24) +
                      (dadoTratado[(TAM_MSG-3)] << 16) +
                      (dadoTratado[(TAM_MSG-2)] << 8)  +
                      (dadoTratado[(TAM_MSG-1)]) ;
                    /* Armazena CRC recebido na serial */
```

Fonte: Elaborado pelo autor.

Em seguida, os dados recebidos são tratados e separados, consoante ao que é mostrado na Figura 40. Cada variável do tipo “dadoX” representa uma informação da serial, como frequência de operação de uma saída PWM ou o valor de resistência de um sensor de temperatura, por exemplo.

Figura 40 – Tratamento de dados

```
uint8_t  dado1  =  dadoTratado[4];
uint8_t  dado2  =  dadoTratado[3];
uint32_t dado3  = (dadoTratado[8]  <<  8) + (dadoTratado[9]);
uint16_t dado4  = (dadoTratado[13] << 16) + (dadoTratado[14] << 8) + (dadoTratado[15]);
```

Fonte: Elaborado pelo autor.

Por fim, são atribuídos pesos distintos aos dados que são somados e comparados ao valor do CRC recebido na serial, conforme indicado na Figura 41 abaixo. Se as informações forem semelhantes, a *flag* é retornada com 1, indicando que a mensagem recebida está correta. Caso contrário, retorna 0, sinalizando algum erro na mensagem.

Figura 41 – Verificação de erro CRC

```
uint32_t CRC_Dados  =  (3 * dado1) + (4 * dado2) + (2 * dado3) + dado4;

if (CRC_Dados == CRC_Serial)
{
 flgDadoCorreto = 1;
}
    else
    {
    flgDadoCorreto = 0;
    }

return flgDadoCorreto;
```

Fonte: Elaborado pelo autor.

3.4 Módulo de resistências

O módulo descrito a seguir tem a função de permitir a emulação de sensores termorresistivos NTC, PTC ou RTD, selecionando um determinado valor de resistência para uma saída. A placa possui três canais independentes, possibilitando a emulação de diversos termistores de forma individual, além da simulação de circuito aberto que será efetuado por uma chave analógica.

Os potenciômetros escolhidos foram o AD5175 de 10 kΩ, 1024 posições, protocolo I2C e três endereços disponíveis e o AD5241 de 1 MΩ, 256 posições e protocolo I2C e quatro endereços disponíveis. Estes dois componentes foram utilizados de forma conjunta, no intuito de obter valores mais próximos do desejado, e lidando com dois potenciômetros para cada canal de emulação de temperatura.

Assim, utilizou-se inicialmente o Potenciômetro 1 (1 MΩ) para ajuste proporcional, obtendo suas informações a partir da Equação (1) (ANALOG DEVICES, 2015):

$$R_{WB}(D) = \frac{D}{256} \times R_{AB} + R_W \quad (1)$$

Onde:

R_{WB} é a resistência ajustada digitalmente (Ω);

D é o número decimal equivalente do código binário (0 – 255);

R_{AB} é a resistência nominal de ponta a ponta do potenciômetro (Ω);

R_W é a resistência da chave que determina a posição do potenciômetro (Ω).

Ambos os potenciômetros utilizam a mesma Equação (1) para determinação da resistência em função da posição digital. Entretanto, visto que o AD5175 possui mais posições $(D\ de\ 0\ a\ 1023)$ e um número de resistência de ponta a ponta menor em relação ao AD5241. Este possui uma melhor resolução, a qual pode ser determinada a partir da Equação (2):

$$RES(Pot) = \frac{R_{AB}}{D_{MAX}} \tag{2}$$

Onde:

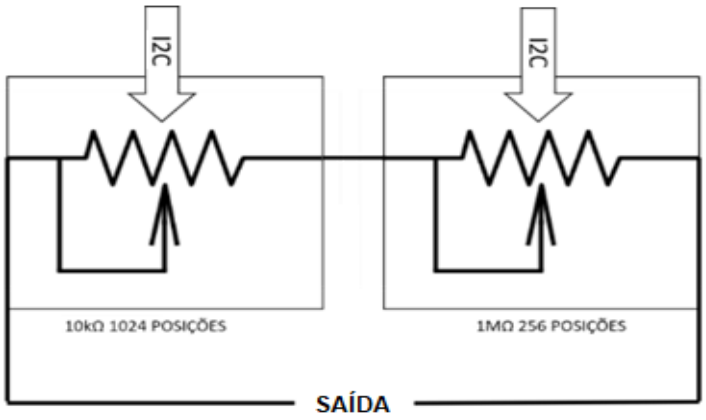
$RES(Pot)$ é a resolução do potenciômetro;

R_{AB} é a resistência nominal de ponta a ponta do potenciômetro (Ω);

D_{MAX} é o número decimal máximo equivalente (255 para o AD5241 e 1023 para o AD5175);

Resolvendo a Equação (2) para as resistências de ponta a ponta respectivas de cada componente, chega-se a uma resolução teórica de 3906,25 Ω/posição para o AD5241 e 9,7656 Ω/posição para o AD5175. Utilizando os potenciômetros em conjunto, resultamos no diagrama apresentado na Figura 42.

Figura 42 – Diagrama de ligação dos potenciômetros digitais



Fonte: Elaborado pelo autor.

Como esses componentes possuem tolerância de 1%, no objetivo de atingir melhores resultados, implementou-se uma etapa de calibração para os potenciômetros. Para tanto, primeiramente, foi determinado experimentalmente o valor de resistência desejado (comandado através da IHM) em relação ao ajustado na prática, a fim de encontrar uma equação característica para seu comportamento, já aplicando a ligação de ambos em série.

Utilizando o multímetro de bancada Minipa MDM-8156B do Laboratório de Instrumentação da Universidade do Vale do Rio dos Sinos, apresentado na Figura 43, foi possível realizar medições mais exatas através do canal de medição a 4-fios.

Figura 43 – Minipa MDM-8156B

Fonte: Elaborado pelo autor.

Uma vez que a faixa de ajuste de resistência é bastante extensa, estas foram separadas em intervalos menores, determinando a equação para cada um para obter maior exatidão em cada faixa, conforme os valores apresentado na Tabela 2 para o *range* de 150 Ω a 1000 Ω.

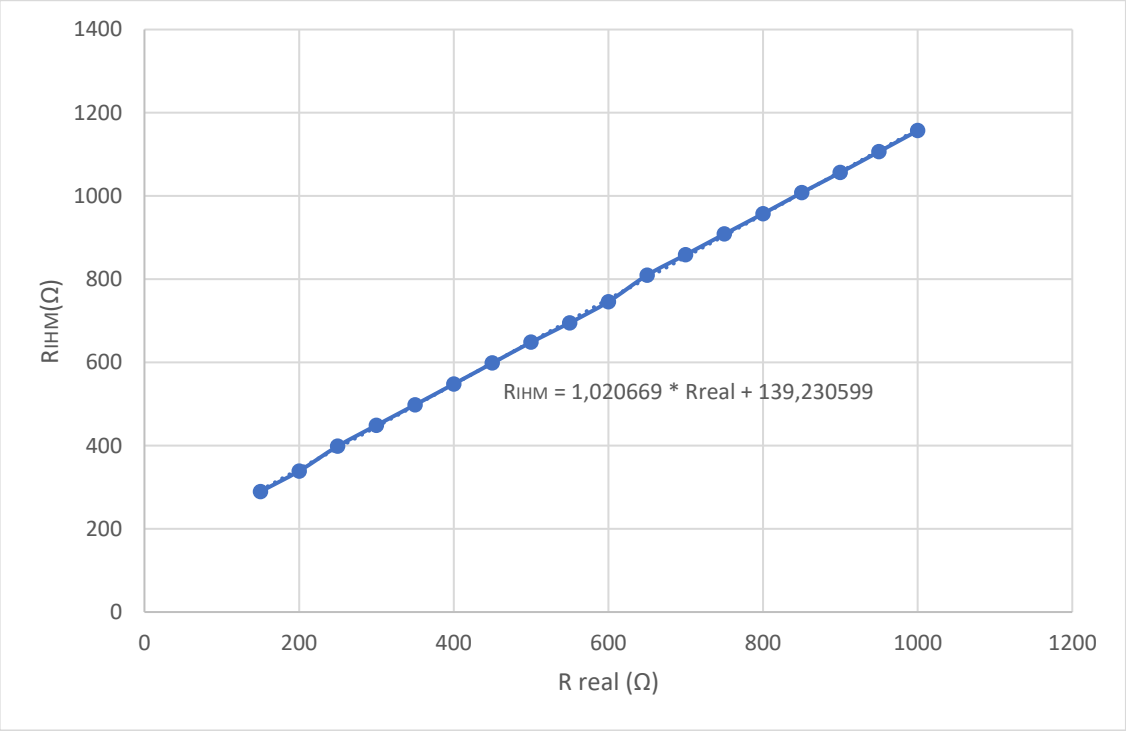
Tabela 2 – Determinação experimental 150 Ω a 1000 Ω.

Resistência IHM (Ω)	**Resistência real (Ω)**
150	289,37
200	338,84
250	398,62
300	448,81
350	497,96
400	547,92
450	598,17
500	648,38
550	695,03
600	745,76
650	809,56
700	858,81
750	908,39
800	957,31
850	1007,74
900	1056,2
950	1106,4
1000	1156,8

Fonte: Elaborado pelo autor.

Gerando a reta fundamentada nos valores apresentados na Tabela 2, obtemos o resultado juntamente a equação característica apresentada na Figura 44. O valor mínimo de resistência possível é de 150 Ω, em virtude de os potenciômetros possuírem por padrão uma resistência na posição 0, e a chave analógica possuir uma resistência de contato ativo.

Figura 44 – Equação da reta característica 150 Ω a 1000 Ω.



Fonte: Elaborado pelo autor.

A medição mostra que o potenciômetro apresenta um erro linear, facilitando a aplicação em software. Este procedimento foi repetido para outras faixas: de 1000 Ω a 10000 Ω, 10000 Ω a 100000 Ω e de 100000 Ω a 1000000 Ω. Em um cenário ideal, poderiam ser aplicadas faixas ainda menores, com o propósito de obter resultados ainda mais exatos. Entretanto, tratando-se de um protótipo, foram avaliados os resultados somente com quatro faixas.

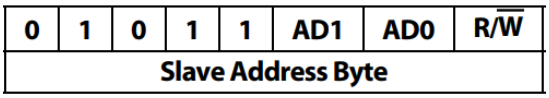
O mesmo método foi replicado para os outros dois canais de resistência, totalizando três canais para este módulo, visto um máximo de três endereços fornecidos pelo potenciômetro AD5175.

Foi possível observar que os dois componentes adotados, possuindo o mesmo fabricante, têm os mesmos endereços disponíveis. Para não haver ocorrência de

conflito na escrita, foi utilizado o barramento I2C1 para o AD5241 e o barramento I2C2 para o AD5175.

As opções de endereçamento para os potenciômetros são apresentadas no *datasheet* dos componentes, mostrados na Figura 45. Os *bits* AD1 e AD0 correspondem ao estado dos pinos no caso do AD5241. Já o AD5175 apresenta apenas um pino ADDR para endereçamento, mas conta com uma lógica auxiliar envolvendo AD1 e AD0 apresentada na Tabela 3, para assim disponibilizar os três possíveis endereços. Por fim, como é realizada somente a escrita, R/W tem o valor fixo 0. (ANALOG DEVICES, 2015), (ANALOG DEVICES, 2010).

Figura 45 – Byte de Endereços AD5241 e AD5175



Fonte: (ANALOG DEVICES, 2015)

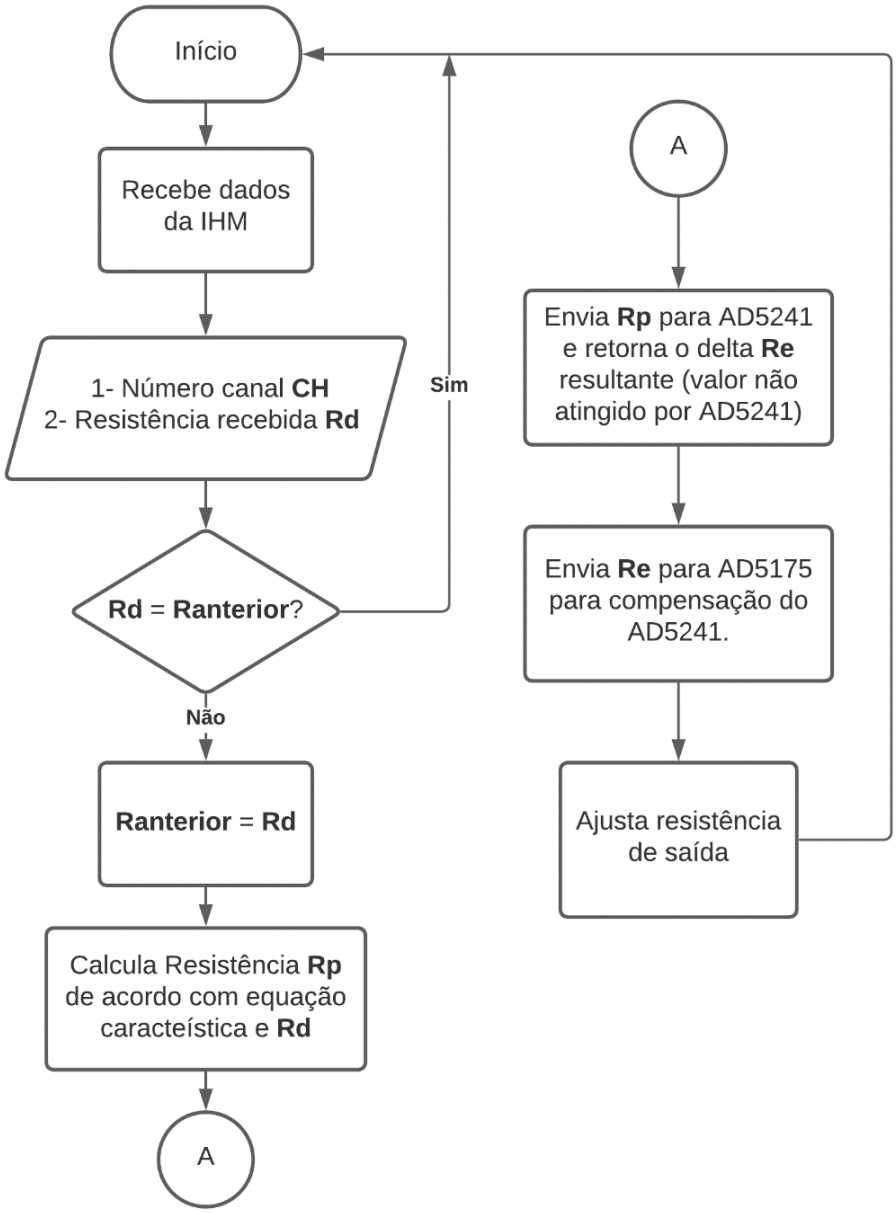
Tabela 3 – Lógica para ADDR AD5175

ADDR Pin	A1	A0	7-Bit I²C Device Address
GND	1	1	0101111
V_{DD}	0	0	0101100
NC (No Connection)[1]	1	0	0101110

Fonte: (ANALOG DEVICES, 2010)

De modo geral, o software desenvolvido para esta aplicação foi estruturado conforme o fluxograma apresentado na Figura 46.

Figura 46 – Fluxograma software módulo de resistências



Fonte: Elaborado pelo autor.

Primeiramente, ambos os potenciômetros são inicializados através das funções “Pot1M_Initialization()” que, conforme mostrado na Figura 47, envia o comando “0x00” referente ao *byte* de instrução e não é utilizado seguido do valor 128, posicionando os potenciômetros exatamente na metade do seu valor máximo para os três endereços de AD5241; e “Pot10k_Initialization()”, que envia o comando 0x1C02, habilitando o potenciômetro para escrita através do Comando 7, conforme o *datasheet* do AD5175, assim como exibido na Figura 48.

Figura 47 – Inicialização AD5241

```
void Pot1M_Initialization()
{
      uint8_t data[]    = {0x00, 128};
      I2C1_Send_data(AD5241_ADDRESS_1, data[0], data[1]);
      I2C1_Send_data(AD5241_ADDRESS_2, data[0], data[1]);
      I2C1_Send_data(AD5241_ADDRESS_3, data[0], data[1]);
}
```

Fonte: Elaborado pelo autor.

Figura 48 – Inicialização AD5175

```
void Pot10k_Initialization()
{/*Envia Comando 7 + Permissão para atualização do potenciometro 5175*/
    uint8_t data[]  = {0x1C, 0x02};
    I2C2_Send_data(AD5175_ADDRESS_1, data[0],data[1]);
    I2C2_Send_data(AD5175_ADDRESS_2, data[0],data[1]);
    I2C2_Send_data(AD5175_ADDRESS_3, data[0],data[1]);
}
```

Fonte: Elaborado pelo autor.

Após a verificação da *flag* de CRC detalhada anteriormente, é calculado a resistência recebida via comunicação serial e armazenada na variável “rxResistance”. O valor desta variável é calculado de acordo com a equação característica encontrada previamente em conjunto com a variável “realResistance”. Os coeficientes são solicitados a partir de uma tabela construída com os valores determinados experimentalmente, de acordo com o número o canal de resistência e faixa de resistência que se deseja operar. Estes coeficientes foram armazenados nas variáveis matrizes “coefA” e “coefB”, mostradas na Figura 49. As linhas correspondem aos três canais de resistência, e as colunas às quatro faixas disponíveis de equações.

Figura 49 – Tabela de coeficientes

```
float  coefA[NUM_CANAIS][NUM_EQS] = {
                                    {1.020669,  1.057102,   1.057902,   1.061090},
                                    {1.027332,  1.063221,   1.064822,   1.073889},
                                    {1.019990,  1.066608,   1.053711,   1.069873}
};

float  coefB[NUM_CANAIS][NUM_EQS] = {
                                    {139.230599, 78.624561, 53.298246,  116.752666},
                                    {143.317269, 87.283746, 66.778901,  128.311123},
                                    {141.011277, 84.223890, 60.419973,  121.478982}
};
```

Fonte: Elaborado pelo autor.

Posteriormente, o resultado deste é cálculo é enviado para o potenciômetro de 1 MΩ realizar o cálculo de posição, como apresentado na Figura 50.

Figura 50 – Código principal Módulo de resistências

```
Pot1M_Initialization();
Pot10k_Initialization();

while (1)
{
ReceiveSerialData();
OrdenaMsgRecebida();
uint8_t CRC_Recebido = CalculoCRC();
    /*Tratamento para sensores de temperatura*/
  if ((CRC_Recebido == 1)&&(dadoTratado[3] == 5))
  {
      rxResistance = (dadoTratado[13] << 16) + (dadoTratado[14] << 8) + dadoTratado[15];
      numeroCanal  = dadoTratado[4] - 1;

      if((rxResistance > 150)    &&    (rxResistance <= 1000))
      {
       realResistance = (coefA[numeroCanal][0]*rxResistance + coefB[numeroCanal][0]);
      }
      if((rxResistance > 1000)    &&   (rxResistance <= 10000))
      {
       realResistance = (coefA[numeroCanal][1]*rxResistance + coefB[numeroCanal][1]);
      }
      if((rxResistance > 10000)   &&   (rxResistance <= 100000))
      {
       realResistance = (coefA[numeroCanal][2]*rxResistance + coefB[numeroCanal][2]);
      }
      if((rxResistance > 100000) &&   (rxResistance <= 1000000))
      {
       realResistance = (coefA[numeroCanal][3]*rxResistance + coefB[numeroCanal][3]);
      }

      realResistance    = rxResistance - (fabs(realResistance - rxResistance));

      if((rxResistance != previousrxResistance) && (rxResistance < 1000000))
      {
          AnalogSwitch_Initialization();
          deltaResistance = Calc_pot1M(dadoTratado[4], realResistance);
          Calc_pot10k(dadoTratado[4], deltaResistance);
          previousrxResistance = rxResistance;
      }
          else if(rxResistance >= 1000000)
          {
              OpenSensor();
              previousrxResistance = rxResistance;
          }
  }

}
```

Fonte: Elaborado pelo autor.

Assim que o cálculo do potenciômetro AD5241 detalhado na Figura 51 é efetuado, utilizando os coeficientes encontrados anteriormente de acordo com o canal de resistência desejado, a diferença entre a resistência recebida da IHM e a resistência calculada é retornada pela função e armazenada na variável “deltaResistance”. Este erro gerado é então enviado para o potenciômetro de 10 kΩ, que possui a função de compensação.

Além disso, se a resistência for superior a 1 MΩ, foi definido que o circuito deve ser posicionado para circuito aberto, invocando a função “OpenSensor()”. Para tal tarefa, selecionou-se a chave analógica 74HC4053 de resistência ativa no contato relativamente pequena (em torno de 80 Ω) e possuindo três canais SPDT (*Single Pole-Double Throw*, ou, do português, polo único duplo acionamento) de controle independente. (NEXPERIA, 2020). A resistência ativa do contato da chave foi considerada na geração da equação característica de ajuste vista anteriormente, a fim de evitar um possível erro.

Figura 51 – Ajuste de posição AD5241 1 MΩ

```
double Calc_pot1M(uint8_t numeroCH, double resistance)
{
  uint8_t  wiperPosition;
  uint16_t AD5241_address;
  double deltaResistance;

  switch(numeroCH)
  {
  case 1: AD5241_address = AD5241_ADDRESS_1;
          break;

  case 2: AD5241_address = AD5241_ADDRESS_2;
          break;

  case 3: AD5241_address = AD5241_ADDRESS_3;
          break;
  }

  if(resistance > (AD5241_MAX_RES / AD5241_MAX_POS)) /*Em caso de a */
  {               /*resistencia desejada for menor que o POT1M pode chegar*/
  /*Calcula a posição do wiper do potenciometro AD5241*/
  wiperPosition = (resistance * AD5241_MAX_POS) / AD5241_MAX_RES;
  /*Calcula o delta entre resistencia desejada e resistencia ajustada*/
  deltaResistance = (wiperPosition *  AD5241_MAX_RES) / AD5241_MAX_POS;
  deltaResistance = resistance - deltaResistance;
  }
  else
  {
     wiperPosition   = 0;
     deltaResistance = resistance;
  }
 /*Envia Byte de Instrução + Posição Potenciometro AD5241*/
 uint8_t data[] = {0x00, wiperPosition};
 I2C1_Send_data(AD5241_address, data[0], data[1]);


 /*Retorna valor de delta para ajuste fino do outro poenciômetro*/
 return deltaResistance;
}
```

Fonte: Elaborado pelo autor.

Enfim, é realizado o cálculo de compensação do potenciômetro AD5175, utilizando os mesmos procedimentos do AD5175, conforme mostrado na Figura 52. Vale ressaltar que cada canal de resistências possui seus respectivos endereços e

valores para equacionamento, que são selecionados ao início das funções de cálculo de posição.

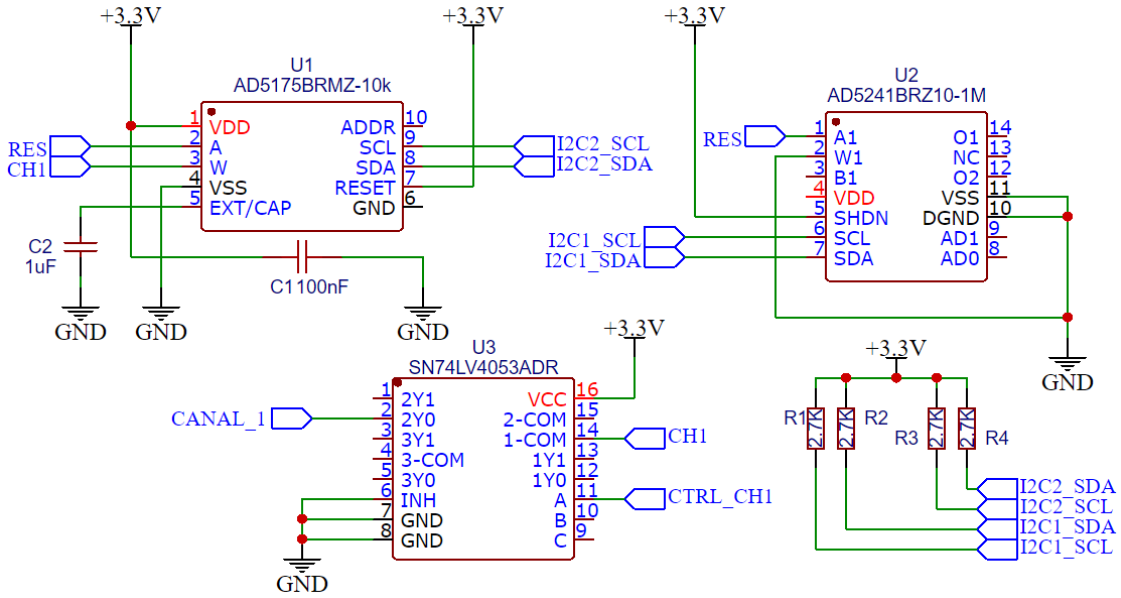
Figura 52 – Ajuste de posição 10 kΩ

```
void Calc_pot10k(uint8_t numeroCH, double resistance)
{
  uint16_t AD5175_address, wiperPosition;
  double erro;

  switch(numeroCH)
  {
  case 1:
        AD5175_address = AD5175_ADDRESS_1;
        break;
  case 2:
        AD5175_address = AD5175_ADDRESS_2;
        break;
  case 3:
        AD5175_address = AD5175_ADDRESS_3;
        break;
  }

  if(resistance > (AD5175_MAX_RES / AD5175_MAX_POS))
  {
    /*Calcula a posição do wiper do potenciometro AD5175*/
    wiperPosition = (resistance * AD5175_MAX_POS) / AD5175_MAX_RES;
    erro = ((resistance * AD5175_MAX_POS) / AD5175_MAX_RES) - wiperPosition;
    if (erro > 0.5) {wiperPosition++;}
  }
  else
  {
     wiperPosition = 0;
  }

  /*Envia Byte de Instrução + Posição Potenciometro AD5175*/
   uint8_t data[]    = {(0x04 /*Comando 1 posicionado no byte*/ +
                                    (wiperPosition >> 8)),
                                     wiperPosition};

  I2C2_Send_data(AD5175_address, data[0], data[1]);
}
```

Fonte: Elaborado pelo autor.

O esquemático do módulo de resistências é apresentado na Figura 53, exibindo a conexão de apenas um canal. Os demais canais possuem estrutura semelhante, variando somente a conexão dos pinos de endereço. A chave analógica também exibe a conexão somente de um canal, disponibilizando mais duas chaves para controle. O pino A do circuito integrado U3 controla a posição da chave 1, isto é, se a resistência está em circuito aberto ou não, enquanto INH deste mesmo componente habilita as chaves para operação quando ligada a GND. O *datasheet* do componente AD5175 recomenda um capacitor de desacoplamento de 100 nF ligado a VDD e o capacitor de 1 uF em EXT/CAP, que é empregado em caso de acesso a memória. Por fim, foram utilizados resistores de 2,7 kΩ para a rede I2C, conforme recomendação do protocolo da rede.

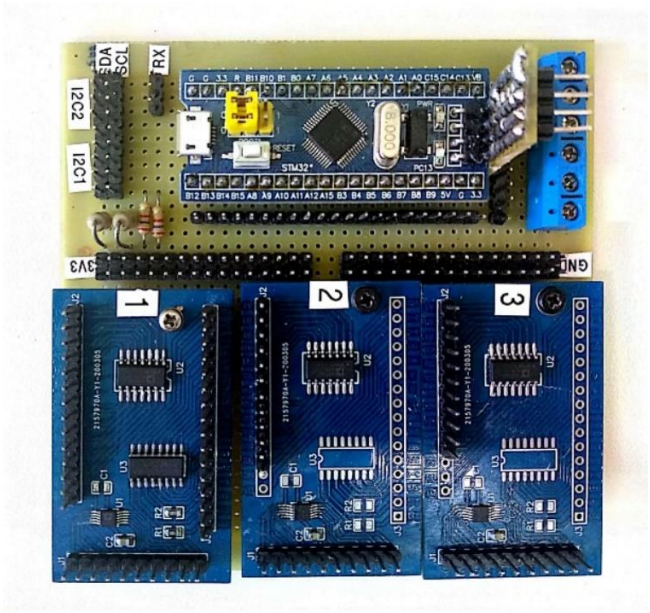
Figura 53 – Esquemático módulo de resistências



Fonte: Elaborado pelo autor.

O hardware foi montado utilizando uma placa de prototipagem, barras de pinos e uma placa adaptador de circuitos integrados SMD (*Surface-mount Technology*, ou, do português, Tecnologia de Montagem Superficial para PTH (*Pin Through Hole*, ou, do português, pino através do buraco*)*, conforme disponível na Figura 54.

Figura 54 – Hardware desenvolvido Módulo de resistências



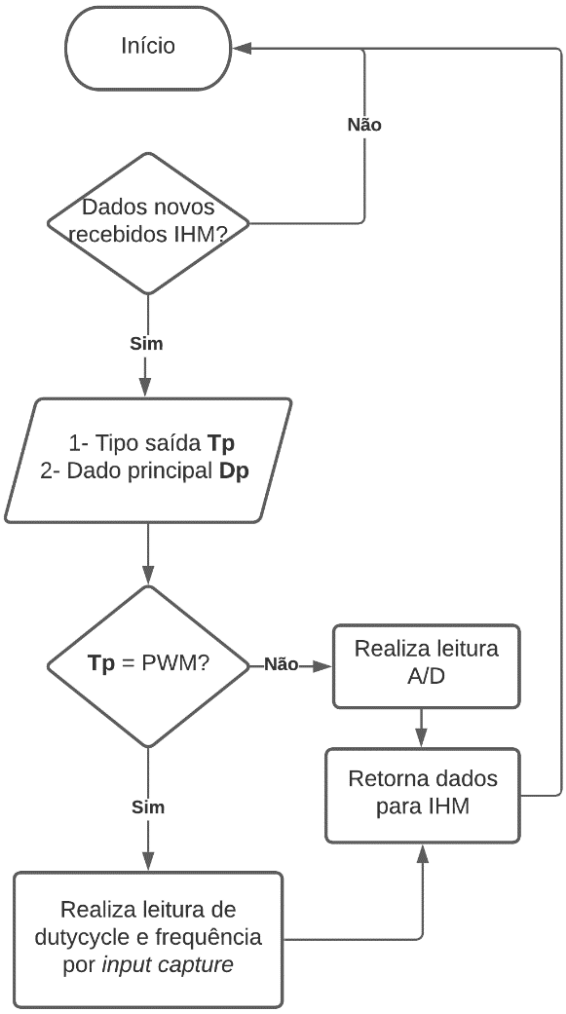
Fonte: Elaborado pelo autor.

3.5 Módulo de entradas PWM/Digitais/Analógicas

A placa desenvolvida tem a função de realizar a leitura das saídas digitais, analógicas e PWM de um determinado produto. As saídas digitais utilizam a mesma configuração da analógica, visto que trata da leitura de uma tensão DC. Esta leitura, por sua vez, se dá mediante a utilização de canais ADC (*Analog to Digital Converter,* ou, em português, Conversor Analógico-Digital) disponíveis do microcontrolador que possui uma resolução de 12 bits. (STMICROELETRONICS, 2015).

O fluxograma do software implementado para leituras das entradas é apresentado na Figura 55.

Figura 55 – Fluxograma software módulo de entradas

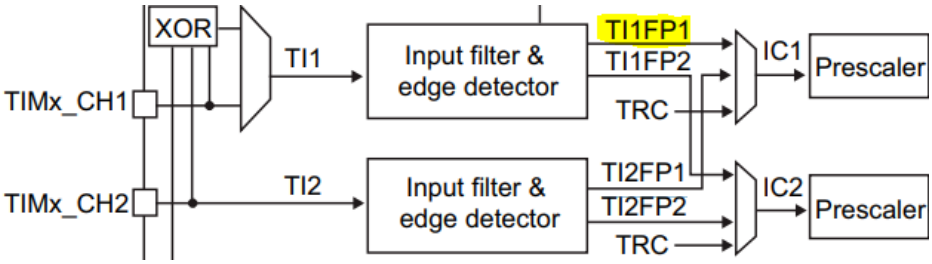


Fonte: Elaborado pelo autor.

3.5.1 Etapa PWM

A leitura de sinais PWM é realizado a partir de uma técnica denominada *Input Capture Mode* (ou, do português, Modo de Captura de Entrada), com a configuração de dois canais de *timer.* O primeiro canal é configurado para detectar o nível de subida da onda, e o segundo canal é configurado para detecção do nível de descida do sinal, juntamente do nível de subida. Para o microcontrolador STM32F103C8T6 é possível realizar esta configuração ajustando *trigger source* para TI1FP1 – a qual define que a fonte para detecção de níveis será referenciada pelo canal 1 (nível de subida configurado). O canal 2 é configurado no modo indireto, compartilhando o mesmo pino de leitura do canal 1, e para nível de descida, permitindo que seja as duas bordas sejam detectadas em um mesmo pino, assim como detalhado na Figura 56 apresentada.

Figura 56 – Configuração TI1FP1



Fonte: Elaborado pelo autor.

Outro fator importante, é a frequência mínima de leitura, que é determinada pelo valor do registrador ARR. Entretanto, o valor de ARR influencia diretamente nas leituras máximas do modo de captura, isto é, quanto maior o valor de ARR, maior a resolução das leituras.

Para tanto, sendo ARR um registrador de 16 *bits*, foi definido o valor máximo, que é de 65536. Assim, a frequência mínima de leitura é calculada utilizando o período do *clock*, dado pelo inverso da frequência de 72 MHz. Este período, que resulta em 13,89 ns, pode ser contado 65536 vezes para detectar uma borda de subida. Dessa forma, devemos multiplicar 13,89 ns por 65536 e obtemos 0,91 ms. Este é o período mínimo a ser detectado e nos fornece, portanto, uma frequência de 1,098 kHz.

Para uma frequência mínima inferior, podemos utilizar o *prescaler* ou PSC para dividir o valor original da frequência de *clock*. No uso de um PSC de 12, a frequência mínima passa a ser de 91 Hz.

A partir destas informações, obtém-se os dados de frequência e *duty cycle* medidos. O código apresentado na Figura 57 mostra como a etapa PWM é executada para um dos canais. Outro detalhe relevante é que, quanto maior a frequência a ser lida, menor o intervalo entre as interrupções, visto que as bordas de subida e descida são detectadas mais rapidamente. Isso pode manter o *software* confinado na interrupção.

Tendo em vista este possível problema, foi criada a variável “countReading1”, responsável por verificar a contagem de leituras. Se esta for superior a 10, a interrupção do *timer* correspondente ao canal lido é temporariamente encerrada. O incremento continua ocorrendo para “countReading1”, ao passo em que o *software* realiza outras tarefas, como a transmissão serial por exemplo. Quando “countReading1” atinge um determinado valor, ela é zerada e a interrupção é ativa novamente, reiniciando a contagem e permanecendo no ciclo de alternância.

Figura 57 – Código Etapa PWM

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    if(htim->Instance == TIM2)
    {
        if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
        {
            IC_R1 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1) + 1 ;

            if(IC_R1 != 0)
            {
                IC_F1 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_2) + 1;
                Frequency1 = ((uint32_t)(2 * HAL_RCC_GetPCLK1Freq())) / 12 / IC_R1;
                dutyCycle1 = (IC_F1 * 100) / IC_R1;
            }

            countReading1++;

            if((countReading1 >= 10) && (flag1 == 1))
            {
                HAL_TIM_IC_Stop_IT(&htim2, TIM_CHANNEL_1);
                HAL_TIM_IC_Stop_IT(&htim2, TIM_CHANNEL_2);
                return;
            }
        }
    }
```

Fonte: Elaborado pelo autor.

Assim, a variável “IC_R1” é responsável por capturar somente as bordas de subida. Então, a cada ciclo de *clock*, a captura conta um. Esta contagem é realizada até que a borda seja capturada, e o mesmo ocorre para a variável “IC_F1”, porém, capturando as bordas de descida e subida.

A frequência é calculada utilizando PCLK1, que informa a frequência do barramento do *timer* correspondente multiplicado por 2 para resultarmos na frequência de *clock* do microcontrolador, dividido pelo PSC e dividido por “IC_R1”. O *duty cycle* é dado pela proporção de um período completo, dado pelo “IC_R1”, e pelo ciclo ativo, dado por “IC_F1”.

3.5.2 Etapa Digital / Analógica

Aa etapa analógica ou digital é realizada utilizando o DMA para leitura simultânea dos canais A/D, conforme mostrado na Figura 58. Assim que ativada, a entrada analógica inicia a sua leitura. O mesmo valor é lido de acordo com o tamanho do filtro definido, para termos uma boa quantidade amostras e posteriormente calculado.

Figura 58 – Código Etapa Digital / Analógico.

```
void EntradasDigAnalog(uint8_t tipoSaida, uint8_t numeroSaida)
{
    uint32_t SOMA_FILTRO = 0;

    for (int i=0; i<TAMANHO_FILTRO; i++)
    {// Leitura das 3 entradas + VREFIN que será valor de referencia
        HAL_ADC_Start_DMA(&hadc1, (uint32_t*)ADC_value, 4);
        SOMA_FILTRO = SOMA_FILTRO + ADC_value[numeroSaida-1];

    }

    SOMA_FILTRO = SOMA_FILTRO / TAMANHO_FILTRO;

    if(SOMA_FILTRO == 0)
    {
    voltage = 0;
    }
    else
    {
    voltage = V_REF / ADC_value[3];
    voltage =  voltage * (float)SOMA_FILTRO;
    voltage = (voltage / RES1) * (RES1 + RES2);
    }
```

Fonte: Elaborado pelo autor.

O cálculo da tensão A/D obtida é baseado na tensão de referência V_REFIN que está disponível para o microcontrolador STM32F103C8T6, possuindo uma tensão fixa de 1,2 V e, portanto, um valor A/D fixo, o qual é lido juntamente das amostras da saída, segundo informação disponibilizada no *datasheet* do microcontrolador,

apresentado na Figura 59. Os resistores de entrada utilizados foram de tolerância 1%, garantindo assim uma medição razoável.

Figura 59 – Parâmetros VREFINT

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{REFINT}	Internal reference voltage	–40 °C < T_A < +105 °C	1.16	1.20	1.26	V
		–40 °C < T_A < +85 °C	1.16	1.20	1.24	
$T_{S\_vrefint}$(1)	ADC sampling time when reading the internal reference voltage	-	-	5.1	17.1(2)	µs
V_{RERINT}(2)	Internal reference voltage spread over the temperature range	V_{DD} = 3 V ±10 mV	-	-	10	mV
T_{Coeff}(2)	Temperature coefficient	-	-	-	100	ppm/°C

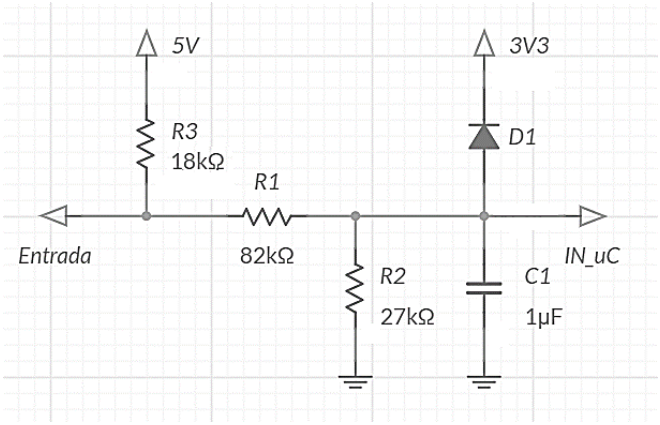
Fonte: (STMICROELETRONICS, 2015)

Os produtos da empresa alvo trabalham com tensões de 12 e 24 VDC. Todavia, devido a fonte de alimentação disponível possuir um limite máximo de 12 V, utilizaremos este valor como extremo. Quando a tensão for de 12 V, é indicado que a tensão de entrada do microcontrolador seja de 3,0 V (a fim de não chegar ao limite de leitura de 3,3 V). Portanto, aplicando a lei de divisores de tensão, com base na Lei de Ohm, podemos obter os resultados desejados.

Adicionalmente, foram incluídas algumas proteções a este circuito de entrada, com o objetivo de evitar problemas de tensões induzidas causados pelo longo comprimento de cabos ou trilhas, que acabam comportando-se como um indutor. Os diodos para proteção ESD (*Electrostatic Discharge* ou, em português, Descarga Eletrostática) são uma forma de grampear os níveis de tensão pretendidos, denominados diodos de *clamping* ou grampeamento. (DIGI-KEY ELECTRONICS, 2012). Foi escolhido o diodo de sinal 1N4148, que se designa por ser de chaveamento rápido (VISHAY, 2019).

Por fim, um filtro passa baixas foi incluído como proteção extra, para evitar ruídos externos de outros circuitos. Como a saída é ativada manualmente, por meio de interação com os produtos, a frequência será muito baixa. Na escolha da frequência de corte, foram colocados valores de capacitância encontrados na própria placa de desenvolvimento. (DIGI-KEY ELECTRONICS, 2012). Os resistores R1 e R2 serão utilizados em conjunto com este capacitor a fim de provocar o efeito desejado. O circuito completo é apresentado na Figura 60.

Figura 60 – Esquemático módulo de entradas



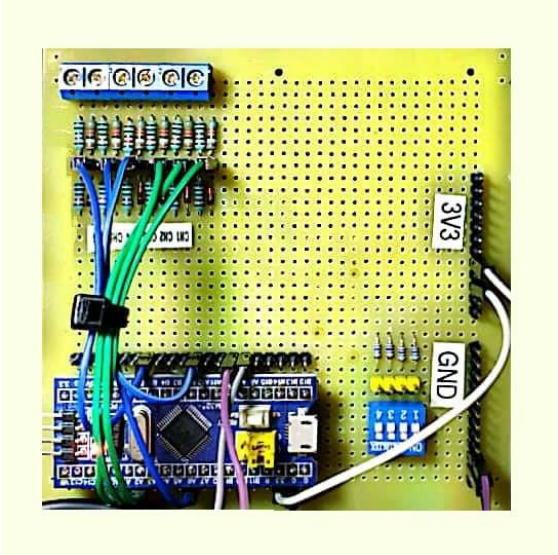
Fonte: Elaborado pelo autor.

Os resistores escolhidos foram R2 de 27 kΩ e R1 82 kΩ. Esta relação nos fornece uma tensão máxima na entrada do microcontrolador, conforme o divisor de tensão, de aproximadamente 2,97 V.

Por fim, o circuito de entrada será comum as configurações analógica ou digital, já o circuito PWM possui entradas específicas que não contém o capacitor C1, uma vez que não é desejado um filtro para este caso.

Semelhante ao módulo de resistências, utilizou-se uma placa de prototipagem para confecção do *hardware,* apresentado na Figura 61.

Figura 61 – Hardware desenvolvido Módulo de entradas

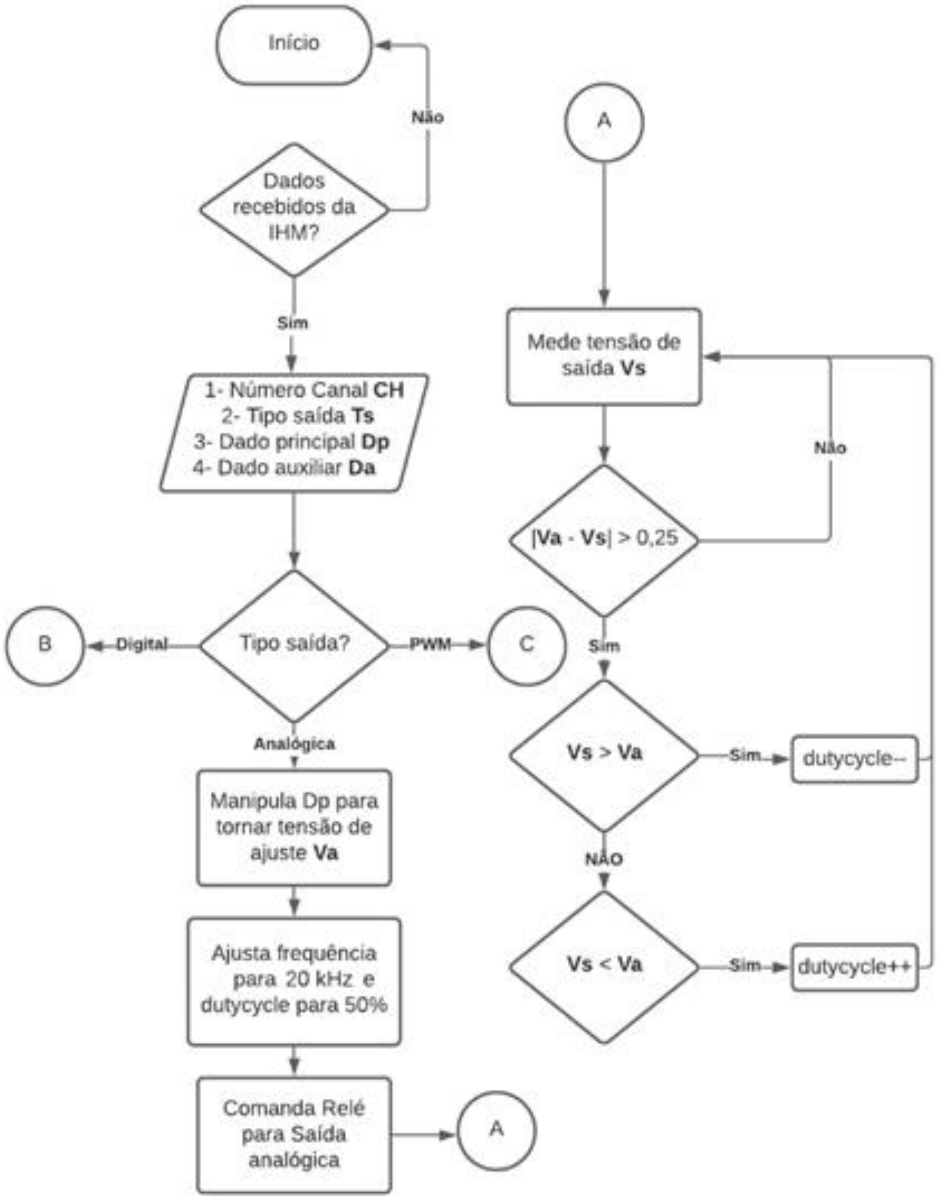


Fonte: Elaborado pelo autor.

3.6 Módulo de saídas PWM/Digitais/Analógicas

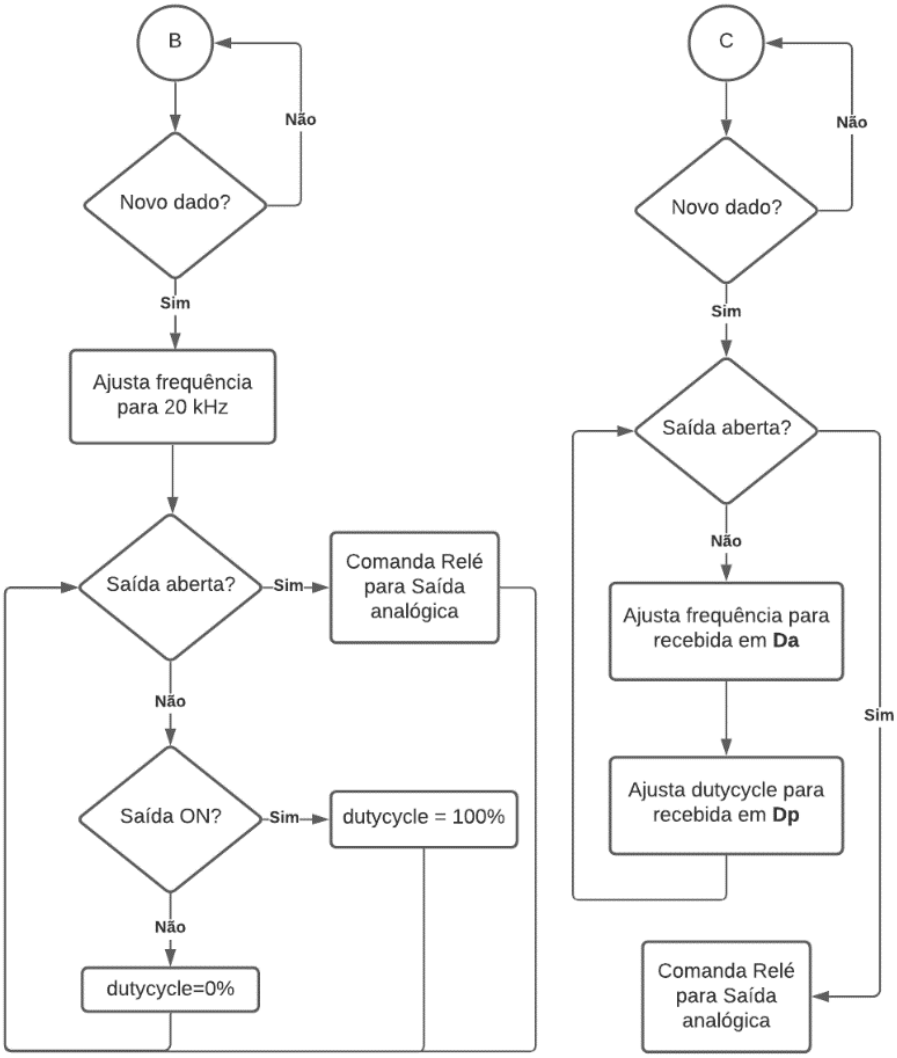
O presente módulo tem por objetivo criar saídas configuráveis em três formatos diferentes: PWM, Digital e Analógica. As configurações estão descritas separadamente nas seções a seguir. Sob uma perspectiva geral, o software executa as tarefas descritas no fluxograma apresentado na Figura 62 e na Figura 63.

Figura 62 – Fluxograma software Módulo de saídas Bloco 1



Fonte: Elaborado pelo autor.

Figura 63 – Fluxograma software Módulo de saídas Bloco 2



Fonte: Elaborado pelo autor.

3.6.1 Etapa PWM / Digital

Este módulo contempla as saídas PWM para validação de entradas PWM de controladores automotivos ou outras aplicações pertinentes. O microcontrolador STM32F103C8T6 dispõe de quatro *timers* para geração PWM (técnica que foi aplicada para obtenção das formas de onda quadrada), cada um com quatro canais. Os canais que compartilham um mesmo *timer* são restritos a frequência deste *timer*, portanto, em um primeiro momento, não é interessante a utilização de mais canais para esta aplicação em específico, posto que visa a possibilidade de configuração individual de frequências. Desta forma, foi utilizado um canal de cada *timer*,

fornecendo-nos quatro saídas de frequência configuráveis e *duty cycles* ajustáveis. (STMICROELETRONICS, 2018).

A partir das informações recebidas da IHM e tratadas via porta serial, obtemos dados do número da saída a ser configurada, categoria, frequência e o *duty cycle*. A relação de resolução entre o ajuste da frequência e *duty cycle* é direta, isto é, no momento em que refinamos a leitura da frequência, deterioramos o duty cycle e vice-versa. Esta relação é dada pelos registradores do microcontrolador manipulados para obtenção da frequência e *duty cycle* desejados.

Para verificar em qual configuração seria suficiente para uma boa exatidão, foram definidos o passo de *duty cycle* em 5% definido pelo registrador ARR. O *duty cycle* varia de 0 % a 100 %, por consequência, se configurarmos ARR em no mínimo 20, teremos que cada valor corresponde a 5%. Para simulação, foi estruturada uma planilha em que os dados de entradas são inseridos e calculados para cada valor de frequência, resultando no valor de PSC necessário e mantendo ARR fixo em 20. A frequência é obtida utilizando a Equação (3).

$$f_{PWM} = \frac{f_{CLK}}{(ARR/PSC)} \tag{3}$$

Onde:

f_{PWM} é a frequência PWM (Hz);

f_{CLK} é a frequência de *clock* (Hz);

ARR é o registrador que define o período;

PSC é o registrador que define o divisor do *clock*;

Varreram-se as frequências de 1 Hz a 100 kHz, calculando os erros teóricos para cada valor. Para um *clock* de 72 MHz, um ARR de no mínimo 20 (a IHM foi ajustada para trabalhar somente com passos de 5%) e um PSC configurável. Mesmo que seja necessário aumentar ARR por PSC chegar ao limite, a IHM ainda manterá os 5% de passo.

Além disso, para uma minimização de erros, foram aplicadas algumas regras também para a frequência de ajuste. A IHM passou a receber um bloqueio para determinadas frequências ainda durante o cadastro, permitindo os valores apresentados na Tabela 4.

Tabela 4 – Definições de frequências

Faixa de frequência (Hz)	Passo (Hz)
1 a 10	1
10 a 1000	10
1000 a 10000	100
10000 a 100000	1000

Fonte: Elaborado pelo autor.

Sob estas condições, os erros obtidos através varredura teórica implementada são apresentados na Tabela 5.

Tabela 5 – Erros varredura de frequências

Erros	Faixa de frequências (Hz)		
	1 a 1000	1000 a 10000	10000 a 100000
de até 10 Hz	0	1	21
de até 100 Hz	0	0	46
de até 1000 Hz	0	0	1
maiores que 1000 Hz	0	0	1

Fonte: Elaborado pelo autor.

Somente no valor de frequência de 96 kHz o erro é superior 1 kHz. O restante das frequências possui um erro teórico de, no máximo, 100 Hz. A partir destas informações, o código do módulo foi desenvolvido como segue na Figura 64.

Cada saída possui um *timer* inteiramente dedicado para sua aplicação PWM, com a informação do número recebido na porta serial. Inicialmente, o código identifica o tipo de saída, definindo-se que o tipo 4 se trata de uma saída PWM. O registrador ARR recebe seu valor de 20, configurando o passo do *duty cycle*, e o valor de PSC é calculado conforme a variável “freqValor”, que recebe a frequência desejada.

Após isso, a variável “dutyCycle” é conferida. Se possuir o valor de 250 significa, conforme definição da IHM, que o circuito necessita “abrir”, então, o relé é comandado a fechar, a saída é aberta, e a *flag*, zerada. Caso contrário, é verificada esta mesma *flag*, denominada “flagSaidaPWMDig1”, que é responsável por evitar que o relé seja acionado indevidamente a cada vez que um novo dado do tipo PWM é recebido. Desta forma, o relé só é acionado uma única vez. Em seguida, a variável “dutyCycle” é comparada a “dutyCycle_anterior1”, configurando o *duty cycle* somente em caso de alguma alteração.

Posteriormente, é acionada uma verificação do PSC. Sendo este valor, por definição do registrador, um inteiro, em caso do cálculo resultar em um número na forma de 36,99, o valor assumido pelo registrador será de 36, visto que ele é programado para truncar este valor. Para resolver isso, é realizado um cálculo no formato decimal (utilizando uma variável *float*) e comparado ao valor inteiro ajustado. Se a diferença entre eles for superior a 0,5, significa que o valor deve ser arredondado para cima e não truncado, sendo isto que o código executa.

Outra proteção presente é o limite de PSC disponível. Sendo um inteiro de 16 bits, o valor máximo que ele pode assumir é de 65536. Para solucionar isto, o algoritmo compensa no valor de ARR, que mesmo possuindo um valor superior, ou seja, um controle de *duty cycle* mais refinado, será mantido ao passo de 5% pela IHM.

Por fim, o código utiliza as funções de atualização dinâmica dos registradores PSC, ARR e CCR (registrador que determina o *duty cycle* em função de ARR), não havendo a necessidade de encerrar e iniciar novamente os *timers* configurados. Vale ressaltar que o código para as saídas digitais é semelhante a este, contudo, mantendo *duty cycle* sempre nos valores extremos de 0% e 100%.

Figura 64 – Código saída PWM / digital

```
if(tipoSaid == 4)
{
uint32_t valorARR  = 20;
uint32_t valorPSC  = ((HAL_RCC_GetSysClockFreq())/(valorARR)/freqValor);

    if(dutyCycle != 250) //250 foi definido para quando o circuito abrir
    {
        if(flagSaidaPWMDig1 == 0)
        {
        HAL_GPIO_WritePin(EN_A1_GPIO_Port, EN_A1_Pin, GPIO_PIN_RESET); //Seleciona PWM
        flagSaidaPWMDig1 = 1;
        }

        if(dutyCycle != dutyCycle_anterior1)
        {
           dutyCycle_anterior1 = dutyCycle;

            erroPSC = (((HAL_RCC_GetSysClockFreq()/valorARR)/(float)freqValor) - 1)
                                                                    - valorPSC;
            if(erroPSC > 0.5)
            {
               valorPSC ++;
            }

            while (valorPSC > 65536)
            {
             valorPSC = ((valorPSC) / 10);
             valorARR = ((valorARR) * 10);
            }

           uint32_t PSC  = valorPSC;
           uint32_t ARR  = valorARR;
           uint32_t CCR  = (valorARR * dutyCycle) / 100;

           __HAL_TIM_SET_PRESCALER(&htim1, PSC-1);
           __HAL_TIM_SET_AUTORELOAD(&htim1, ARR-1);
           __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, CCR-1);
        }
    }

    else //Em caso da solicitação for para abrir o circuito
    {
        if(flagSaidaPWMDig1 == 1)
        {
         dutyCycle_anterior1 = dutyCycle;
         HAL_GPIO_WritePin(EN_A1_GPIO_Port, EN_A1_Pin, GPIO_PIN_SET);
         flagSaidaPWMDig1 = 0;
        }
    }
}
```

Fonte: Elaborado pelo autor.

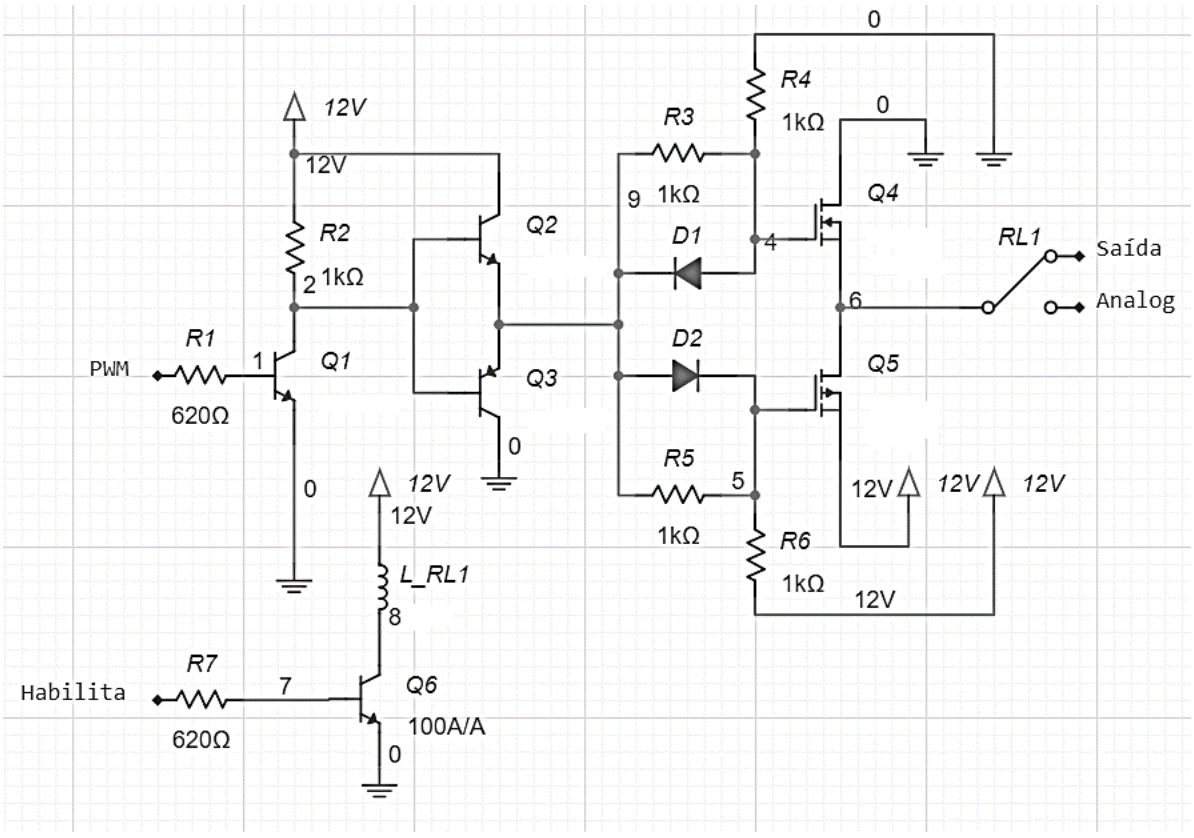
A partir daqui, a etapa de software está concluída. O circuito PWM implementado baseou-se na configuração *push-pull* com *enable,* ou simplesmente um circuito *tristate,* com algumas modificações que se fizeram fundamentais para atender as necessidades da aplicação.

Os dados para saída digitais possuem exatamente a mesma codificação, porém, utilizando uma frequência fixa de 20 kHz e o valor de *duty cycle* de 0% que corresponde a *OFF* e 100% que corresponde a *ON.*

Havendo a necessidade de um circuito capaz de lidar com uma frequência de até 100 kHz, foi escolhido o MOSFET (*Metal-Oxide-Semiconductor Field-Effect Transistor*, do português, Transistor de Efeito de Campo de Óxido de Metal Semicondutor) IRF7343PbF, disponibilizado pela empresa alvo, que possui o encapsulamento de circuito integrado e se trata de um MOSFET duplo, possuindo um tipo N e um tipo P internos. (INFINEON TECHNOLOGIES, 2020).

Observou-se, de forma experimental, que o MOSFET é bastante sensível. O tipo P desliga somente com uma tensão de *gate* perto 12 V e o tipo N com uma tensão de *gate* próxima de 0 V. Logo, o circuito de habilita que realiza a função de circuito aberto foi implementado através de um simples relé, como é possível observar no circuito apresentado na Figura 65.

Figura 65 – Circuito Saída PWM / Digital



Fonte: Elaborado pelo autor.

Quando o sinal PWM ou digital está em nível lógico alto, os transistores Q1 e Q3 estarão polarizados. Com isso, teremos 0 V nos emissores de Q2 e Q3. Assim, no *gate* de Q5 teremos 6 V pelo divisor de tensão R5 e R6 e o MOSFET Q4 estará desligado, com 0 V no seu *gate*. Portanto, teremos 12 V na saída. O circuito *push-pull* composto por Q2 e Q3 tem a função de equalizar a tensão de polarização dos MOSFETs.

Em contrapartida, quando o sinal PWM ou digital está em nível lógico baixo, o transistor Q1 estará desligado e a tensão de 12 V proveniente do resistor de *pull-up* estará na base do circuito *push-pull* auxiliar, e Q2 conduzindo. Com isso, teremos 6 V na pelo divisor de tensão R3 e R4 no *gate* de Q4 e Q5 permanecerá desligado. Desta maneira, teremos 0 V na saída.

O sinal de habilita, quando ativo, mantém a saída com os sinais de saída do MOSFET. Caso esteja inativo, o relé RL1 estará desligado e o sinal PWM estará comutado para o circuito analógico, ao mesmo tempo mantendo a saída PWM / Digital em estado aberto.

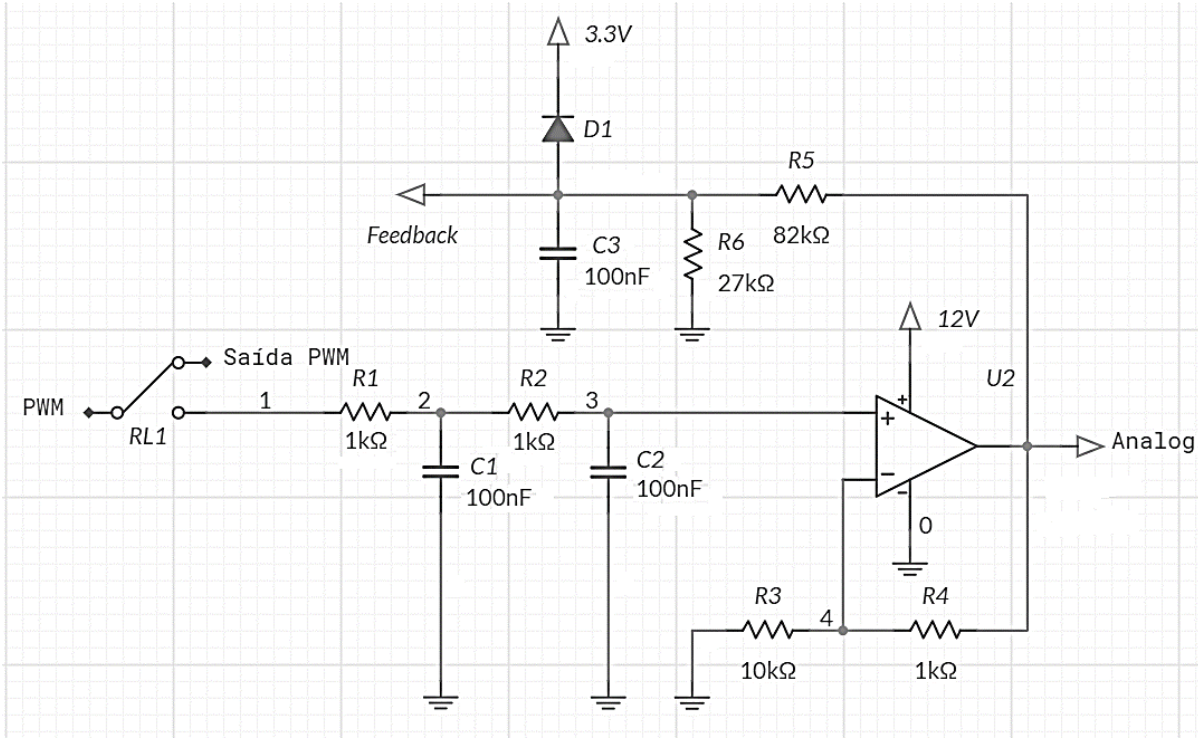
3.6.2 Etapa Analógica

Este módulo tem a função de testar as entradas analógicas de um determinado produto, contendo saídas de tensão DC ajustável. Para isto, foi empregado o mesmo circuito PWM aplicado anteriormente. Ao mesmo tempo em que o relé possui a função de abrir a saída “PWM”, também é utilizado para comutar a saída analógica. Entretanto, a saída, desta forma, poderá ser ou PWM/digital ou analógica.

A saída PWM, ao ser comutada para o circuito analógico, passa por um filtro responsável por eliminar as harmônicas indesejadas e manter somente a componente DC. Em seguida, o sinal resultante passa por uma etapa de ajuste de ganho mediante um amplificador não inversor. Enfim, a saída ajustada é medida pelo microcontrolador para novos ajustes, sendo a tensão analógica regulada em função do *duty cycle* da onda PWM e conforme parâmetros do filtro.

O cálculo do filtro é apresentado no APÊNDICE C . O filtro foi calculado para uma frequência dez vezes menor que a frequência do PWM, isto é, em torno de 2 kHz para garantirmos um nível DC sem ondulações na saída, e utilizando dois polos. Já o APÊNDICE D – Cálculo amplificador não-inversor mostra o cálculo do amplificador não inversor obtido, com ganho definido em no máximo 2.

Figura 66 – Circuito Saída analógica



Fonte: Elaborado pelo autor.

Pode ser observado na Figura 66 que o amplificador não-inversor foi ajustado para um ganho aproximadamente igual a 1, dado que ganhos muitos superiores manteriam a tensão analógica gerada sempre no valor máximo para grande parte dos *duty cycles*. A entrada de retorno ou *feedback* foi calculada de forma a levar a 12 V a um valor próximo de 3 V, utilizando resistores de 5% de tolerância que estavam disponíveis, no intuito de ajustar o nível de tensão para uma faixa suportada pelo microcontrolador. O diodo de *clamping* e o capacitor foram inseridos pelo mesmo propósito apresentado no módulo de entradas. Por fim, o amplificador disponibilizado para confecção deste módulo foi o LM358, que é um componente barato e bastante comum em aplicações que exigem operacionais.

A Figura 67 apresenta a etapa do código referente a saída analógica. Utiliza a mesma lógica de *flag* para o relé, porém, de forma inversa a PWM/digital. As variáveis “voltage” e “analogVoltage” verificam se é um novo valor a ser ajustado ou se é semelhante ao anterior, garantindo que a função de ajuste AD seja acionada somente quando necessário.

Figura 67 – Código saída analógica

```
 if(tipoSaid == 3)
{
  __HAL_TIM_SET_PRESCALER(&htim1, 0);
  __HAL_TIM_SET_AUTORELOAD(&htim1, 3600-1); /*Ajusta frequencia em 20kHz*/

  if((dutyCycle != 250) && (voltage >= 0) && (voltage <= 12))
  {
    if(flagSaidaAnalog1 == 0) /*Seleciona Analógica*/
    {
      HAL_GPIO_WritePin(EN_A1_GPIO_Port, EN_A1_Pin, GPIO_PIN_SET);
      flagSaidaAnalog1 = 1;
    }

    if(voltage != analogVoltage)
    {
      analogVoltage = voltage;
      AjusteAD1(numSaid, voltage, 0);
    }
  }

  else /*Seleciona PWM para manter aberto*/
  {
    if(flagSaidaAnalog1 == 1)
    {
    analogVoltage = voltage;
    HAL_GPIO_WritePin(EN_A1_GPIO_Port, EN_A1_Pin, GPIO_PIN_RESET);
    flagSaidaAnalog1 = 0;
    }
  }
}
```

Fonte: Elaborado pelo autor.

A Figura 68 mostra a função de ajuste A/D, que está diretamente relacionada a saída analógica. Assim que ativada, a saída analógica inicia a sua leitura com uma entrada de retorno para o microcontrolador. A leitura A/D é realizada também utilizando DMA, por motivos já discutidos anteriormente. O mesmo valor é lido dez vezes, para termos uma boa quantidade amostras, e então calculado.

O cálculo da tensão A/D obtida é baseado na tensão de referência V_REFIN, semelhante ao controle já descrito para o módulo de entradas.

Uma vez calculada, a tensão de entrada do A/D é comparada a tensão desejada, e o *duty cycle* da onda PWM ajustado até que este erro seja o mínimo possível.

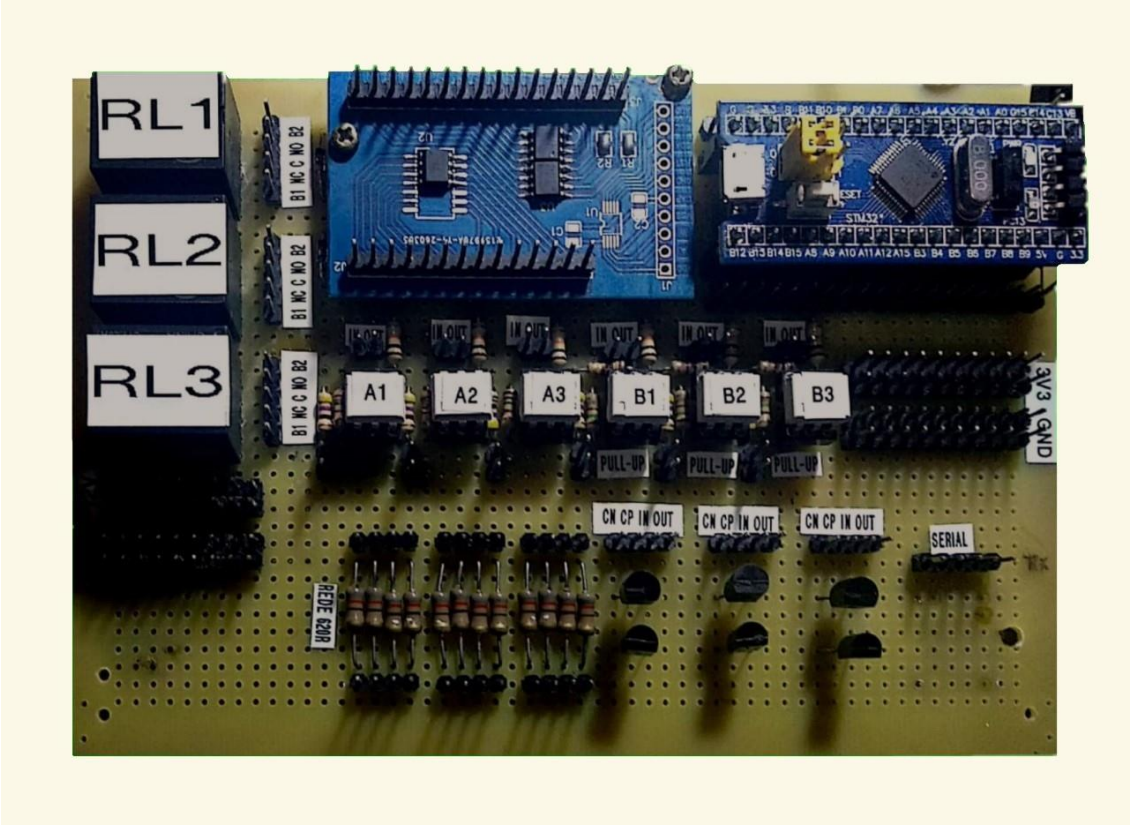
Figura 68 – Ajuste A/D saída analógica

```
while(voltageCompare > 0.02) //erro máximo definido em 0.05V
{
    for (int i=0; i<TAMANHO_FILTRO; i++)
    {
    HAL_ADC_Start_DMA(&hadc1, (uint32_t*)ADC_value, 4);
    // Leitura das 3 entradas + VREFIN que será valor de referencia
    SOMA_FILTRO[0] = SOMA_FILTRO[0] + ADC_value[numero-1];
    }
    SOMA_FILTRO[0] = SOMA_FILTRO[0] / TAMANHO_FILTRO;

    if(ADC_value[numero-1] == 0)
    {
        voltageADC = 0; //protecao para valor tendendo a infinito
    }
    else
    {
    voltageADC = (1.2 / ADC_value[3]);
    voltageADC =  voltageADC * (float)SOMA_FILTRO[0];
    voltageADC = (voltageADC / RES1) * (RES1 + RES2);
    }

    voltageCompare = fabs(voltageADC - voltage); //verifica erro

       if(voltageADC < voltage)
       {
           if (CCR >= 3599)
           {
               CCR  = 3599;
           }
           else
           {
               CCR++;
           }
       }
       if(voltageADC > voltage)
       {
           if (CCR <= 0)
           {
               CCR = 0;
           }
           else
           {
               CCR--;
           }
       }
```

Fonte: Elaborado pelo autor.

A Figura 69 apresenta o *hardware* desenvolvido para esta aplicação. Utiliza as mesmas placas adaptadoras SMD para PTH para posicionamento dos MOSFETS.

Figura 69 – Hardware desenvolvido Módulo de saídas



Fonte: Elaborado pelo autor.

O módulo proposto, além disso, foi todo confeccionada através de uma placa de prototipagem, de forma semelhante ao módulo de resistências.

4 RESULTADOS E DISCUSSÕES

4.1 IHM

A IHM foi desenvolvida de modo a funcionar como um sistema operacional. Diversas tarefas são executadas em intervalos e prioridades definidas. A conexão é realizada à uma taxa de 19200 kbps, apresentando eventuais travamentos momentâneos, ou simplesmente uma demora de resposta. Foi observado que para algumas taxas de transmissão, como 9600 kbps e 115200 kbps, esta não operou corretamente. Também pôde ser observado que quanto maior o volume de informações presente na rede, menor a velocidade de resposta da IHM.

A IHM permite realizar a configuração e a interação com o teste conforme a especificação proposta, possibilitando que o usuário valide um controlador automotivo com maior facilidade.

4.2 Módulo de resistências

4.2.1 Medições

Existem outras opções mais sofisticadas nas quais este módulo poderia ter sido desenvolvido, contudo, optou-se pela de menor custo. Duas possíveis soluções seriam utilizar uma topologia com multiplexadores ou relés de estado sólido, controlando via software a saída e a seleção de entradas que apresentariam, por sua vez, resistores em série. Esta técnica é utilizada nas placas comerciais apresentadas na fundamentação do presente trabalho.

Em contrapartida, devido ao grande volume de componentes necessários e o custo envolvendo este método (principalmente por parte dos relés de estado sólido), elegeram-se os potenciômetros digitais, que não são usuais neste tipo de aplicação e mostram-se como uma oportunidade de verificar esta alternativa de menor custo.

Se tratando dos potenciômetros digitais, também teríamos outras alternativas para obtermos melhores resultados. Uma vez que a faixa de ajuste é muito extensa (0 Ω a 1 MΩ), foram utilizados dois divisores de tensão combinado a um A/D de 12 bits do microcontrolador para calibração, juntamente à chave analógica que possui a

função de abrir o circuito, aproveitando para selecionar o divisor. Um conversor A/D de 12 bits fornece uma resolução de em torno de 0,8 mV.

Esta técnica não funcionou de forma adequada para valores cuja resistência é distante do resistor fixo do divisor de tensão, apresentando um erro considerável. Assim, à medida que o valor a ser calibrado se torna mais distante, menor a exatidão no valor final e, por consequência, menor a confiabilidade na resistência ajustada.

A solução final seriam diversos divisores de tensão em combinação com chaves analógicas diferentes para cada faixa de medida, porém, a calibração adotada na metodologia acabou por ser uma alternativa mais prática e de menor custo, visto que não necessita componentes auxiliares ao se basear somente no comportamento do próprio potenciômetro.

A Tabela 6 apresenta os resultados obtidos para o módulo de resistências para valores gerados em diferentes faixas, com medições realizadas em triplicatas (apresentando somente a média destes valores) e erro médio a partir destas mesmas medições. O instrumento utilizado para esta tarefa foi o MDM-8156B, apresentado na Figura 43. Observa-se que para valores menores de resistência, que foram calibrados em uma faixa mais estreita, foram obtidos resultados melhores.

Tabela 6 – Resultado entradas de resistência

Resistência IHM (Ω)	**Resistência média ajustada (Ω)**	**Erro médio (Ω)**
150	149,19	±0,66
250	249,61	±0,24
500	498,20	±1,55
1000	998,01	±2,65
5500	5490,8	±9,51
10000	9980,7	±21,27
22000	21980,2	±19,78
120000	119300	±703,75
500000	498700	±1312,56
750000	754000	±5995,64

Fonte: Elaborado pelo autor.

4.2.2 Influência da temperatura

Em acréscimo a este desenvolvimento, os potenciômetros foram verificados em um teste de temperatura, condicionando os componentes à um ciclo térmico. Para

tanto, foi empregada a câmara térmica disponível no Laboratório de Testes da empresa alvo. É disponibilizado pelo datasheet que o AD5241 e o AD5175 possuem um coeficiente de temperatura de 30 ppm/ºC e 35 ppm/ºC, respectivamente. (ANALOG DEVICES, 2015) (ANALOG DEVICES, 2010).

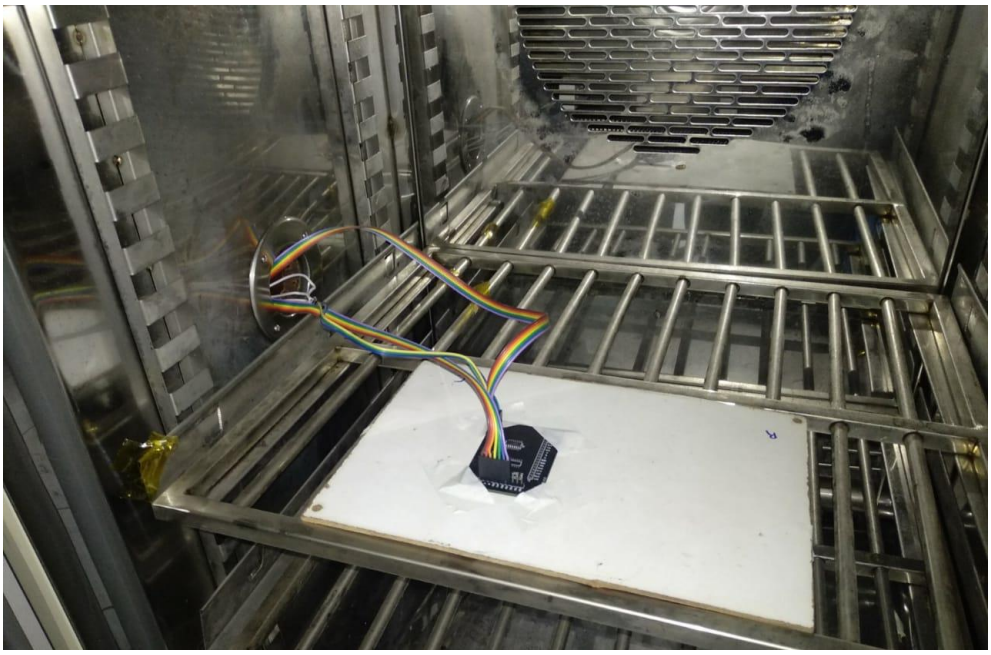
Desta forma, calcula-se que para o potenciômetro de 10 kΩ temos uma variação máxima de 0,35 Ω/ºC (considerando o valor de 10 kΩ). Já o potenciômetro de 1 MΩ, também considerando seu valor máximo, teríamos uma variação de 30 Ω/ºC. (RIEDON RESISTOR MANUFACTURING, 2014).

Desenvolvido para operar em ambiente de laboratório, a faixa de temperatura ajustada foi de 0 ºC por 20 minutos e, em seguida, de 50 ºC por outros 20 minutos. A Figura 70 mostra as duas configurações ajustadas na câmara, e a Figura 71 apresenta os potenciômetros inseridos na câmara térmica.

Figura 70 – Ajustes de temperatura na câmara térmica

Fonte: Elaborado pelo autor.

Figura 71 – Potenciômetros digitais inseridos na câmara térmica



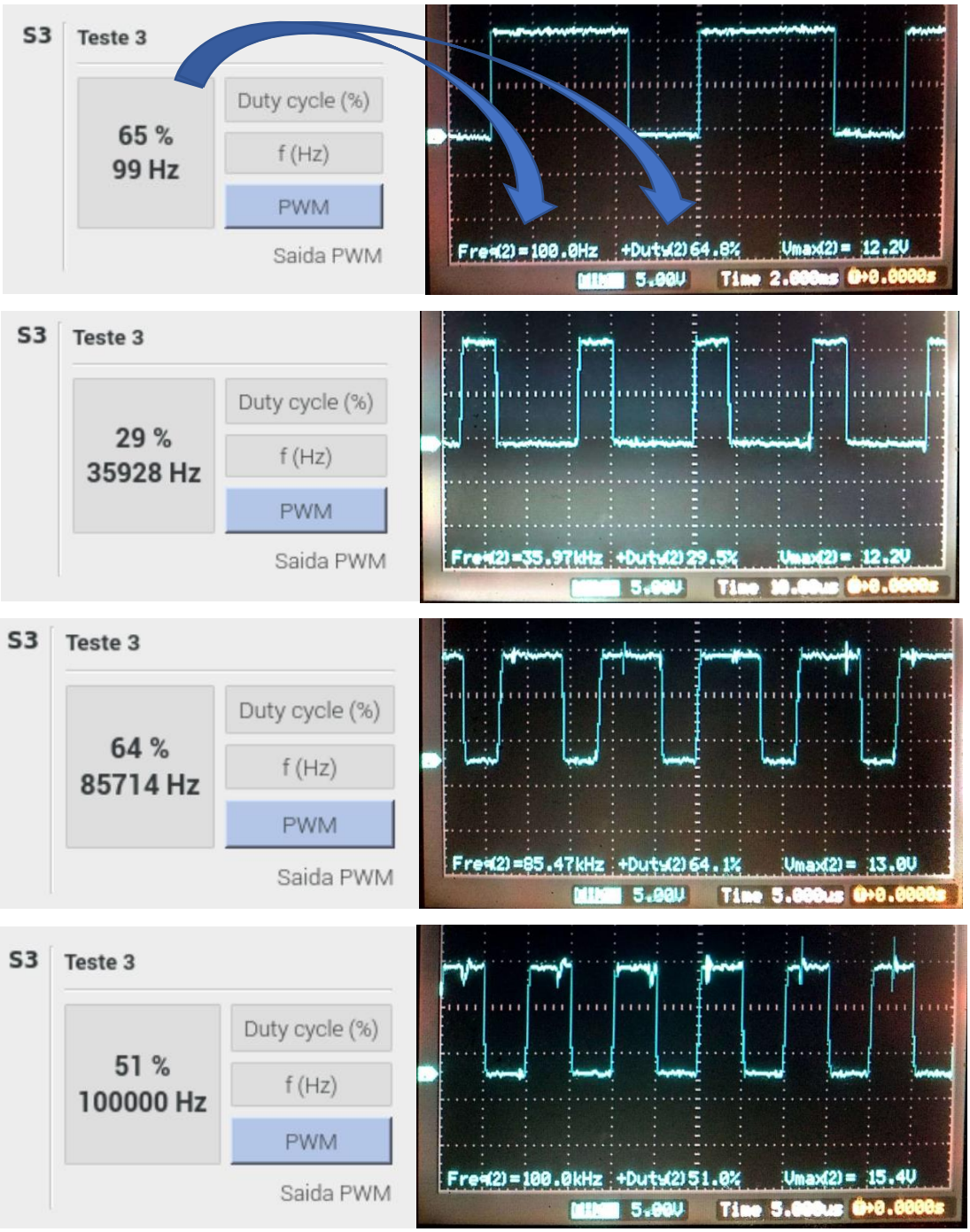
Fonte: Elaborado pelo autor.

Verificou-se que em torno de 50 ºC o potenciômetro de 1 MΩ variou em torno de 157 Ω. Já o de 10 kΩ, variou em torno de 6,3 Ω. Considerando o ambiente de laboratório com temperatura controlada, estas variações serão muito menores e não impactarão significativamente na performance do módulo.

4.3 Módulo de entradas

A presente seção explana os resultados obtidos para a leitura de entradas nas três configurações implementadas: PWM, digital e analógica. A Figura 72 exibe os resultados atingidos para leitura de frequências e *duty cycle* em diversas faixas, utilizando um gerador de funções para produzir os sinais de onda quadrada. Nota-se que frequências com valores não usuais, como 35,97 kHz e 85,47 kHz, apresentam erros de leitura de 70 Hz e 240 Hz, respectivamente. Este erro tende a aumentar conforme o aumento da frequência, dado um menor número de amostras para contagem. Entretanto, de modo geral, os resultados se demonstraram satisfatórios, uma vez que 85,47 kHz, um valor muito próximo do limite máximo de leitura estipulado, apresentou erro inferior a 300 Hz.

Figura 72 – Resultado das entradas PWM



Fonte: Elaborado pelo autor.

Para as saídas analógicas e digitais, foi aplicada uma fonte de tensão externa, com o propósito de ajustar diferentes valores. A medida realizada pelo módulo foi comparada a um multímetro Fluke 110 Plus True RMS, disponibilizado pela empresa alvo.

Figura 73 – Resultado das entradas analógicas

Fonte: Elaborado pelo autor.

A medição realizada pela entrada analógica é apresentada na Figura 73 e se mostrou bastante próxima da saída real. O multímetro apresenta a incerteza de medição própria, todavia, a leitura apresenta erros inferiores a 0,2 V, suficientes para as aplicações de teste desejadas.

Por fim, para validação das entradas digitais, foram utilizadas as mesmas tensões das entradas analógicas. O comportamento discreto funcionou corretamente para os três estados possíveis. O resistor de pull-up ligado a 5 V opera quando não há sinal conectado. Desta forma, o *software* interpreta que o circuito está aberto para tensões inferiores a 8 V e superiores a 2 V. A Figura 74 apresenta os resultados obtidos para as entradas digitais, que se demonstraram conforme esperado, apresentando os três estados desejados.

Figura 74 – Resultado das entradas digitais

Fonte: Elaborado pelo autor.

4.4 Módulo de saídas

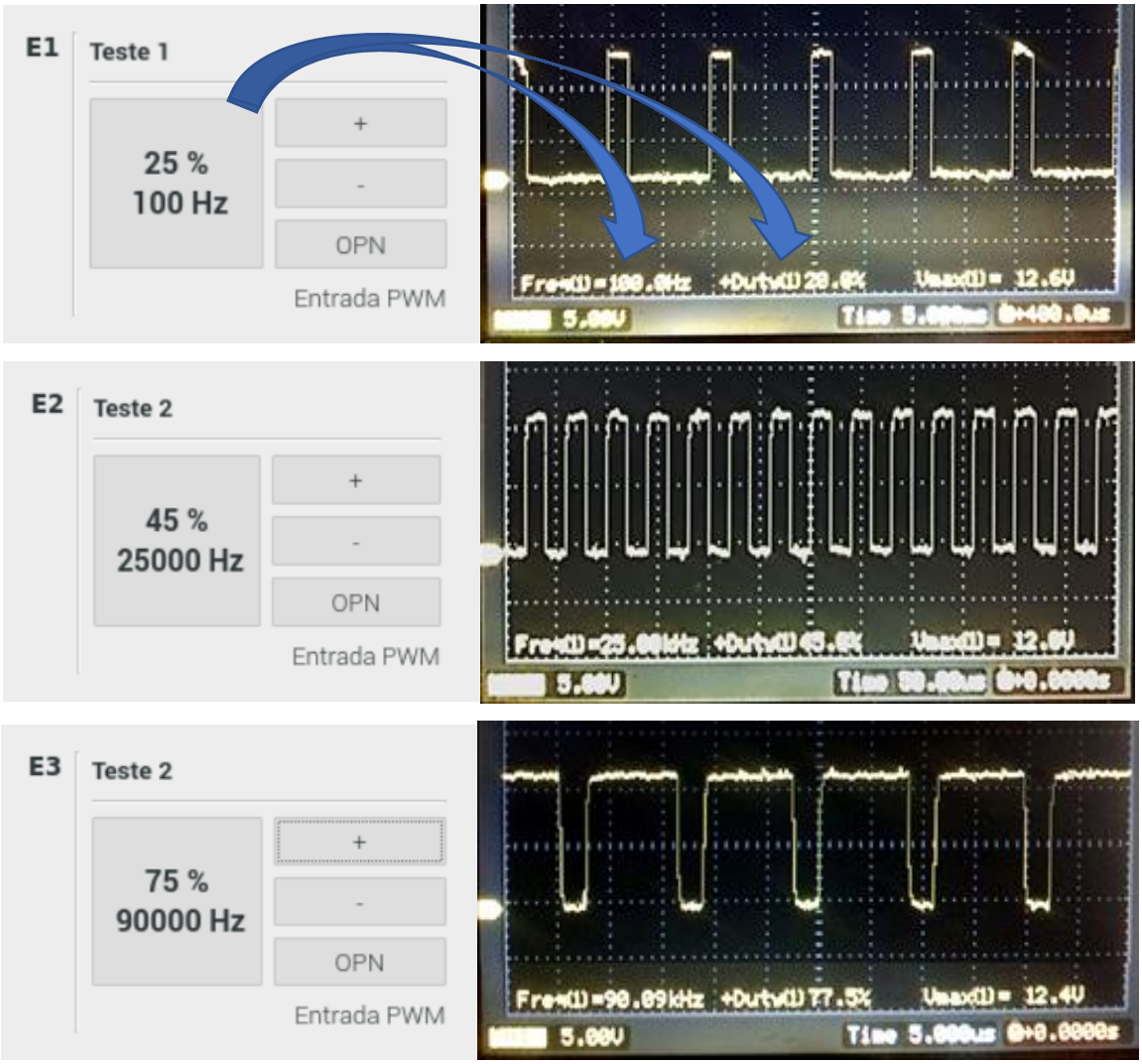
Se tratando de um protótipo, não foram implementadas proteções para o circuito. Em caso de uma conexão equivocada na saída, o MOSFET provavelmente aqueça e seja danificado. É possível acrescentar uma *polyswitch*, que possuiria a função exclusiva de proteção contra sobrecorrente da saída. Estes componentes aumentam sua resistência com a elevação da temperatura, até que o circuito seja “aberto”. Assim que os níveis de corrente normalizam, ele retorna a sua resistência normal. (LITTELFUSE, 2020).

Outra melhoria seria a substituição dos transistores BJT (*Bipolar Junction Transistor)* por opto-acopladores, a fim de isolar galvanicamente os pinos do

microcontrolador. Entretanto, não havia a disponibilidade de componentes capazes de chavear frequências elevadas como a de 100 kHz e optou-se, então, pela utilização dos transistores.

O módulo de saídas foi avaliado em suas três configurações: PWM, digital e analógico. A Figura 75 apresenta medições realizadas para diferentes frequências e *duty cycles*. O osciloscópio mostra a medida real efetuada, enquanto a IHM mostra os valores desejados.

Figura 75 – Resultado das Saídas PWM

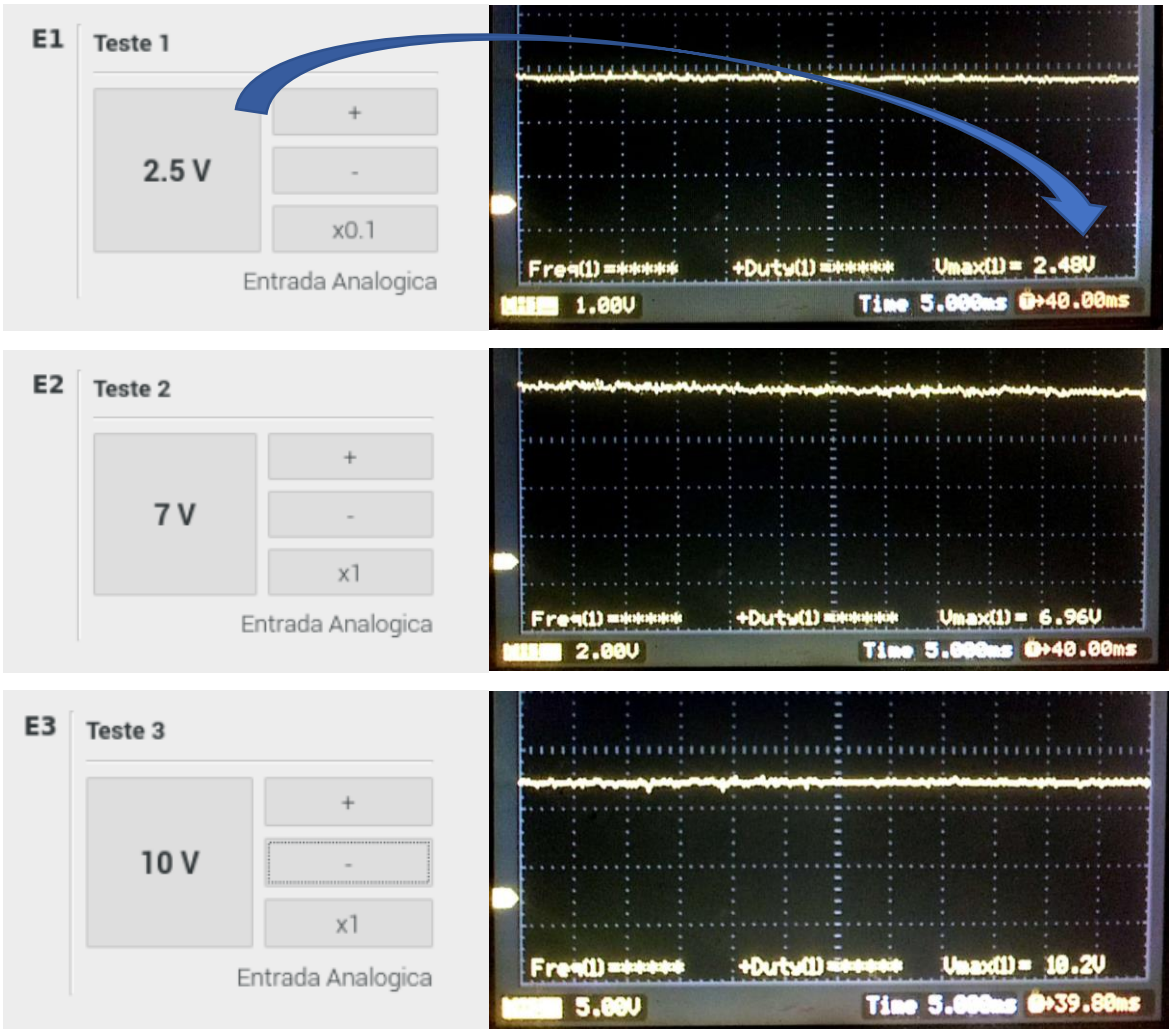


Fonte: Elaborado pelo autor.

As saídas se ajustam corretamente conforme comandos executados na IHM, e é possível perceber que tanto frequência, quanto *duty cycle,* apresentam os valores esperados.

A Figura 76 retrata os resultados para as saídas analógicas. A tensão atingida pelas saídas com a realimentação proposta conseguiu manter valores próximos do alvo, mas, ainda apresentando alguns erros em virtude dos divisores de tensão utilizando resistores de 5% de tolerância.

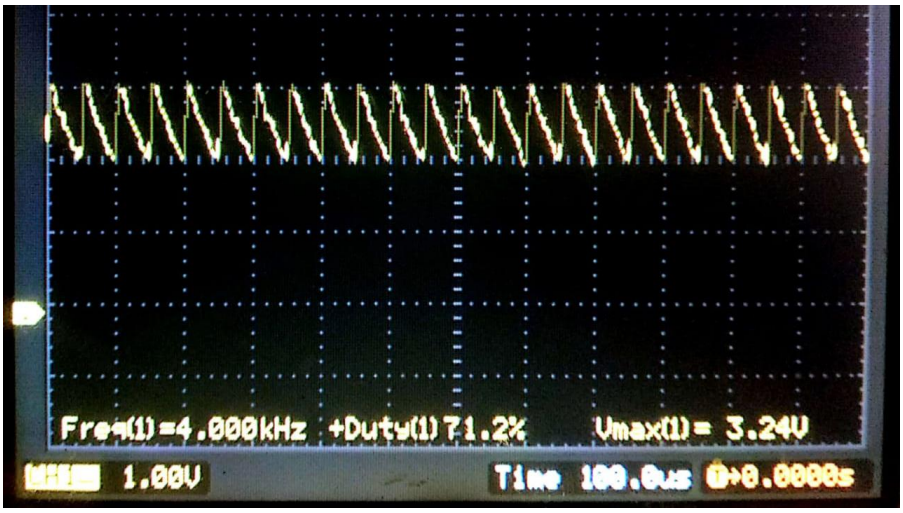
Figura 76 – Resultado das Saídas analógicas



Fonte: Elaborado pelo autor.

Entretanto, foi observado também para esta configuração analógica a presença de uma ondulação, exibida na Figura 77, provavelmente em decorrência do filtro calculado para uma frequência de corte de 2 kHz. Seria possível diminuir ainda mais a frequência de corte a fim de eliminar esta harmônica parasita, porém, este problema foi detectado somente após o término da confecção do *hardware*.

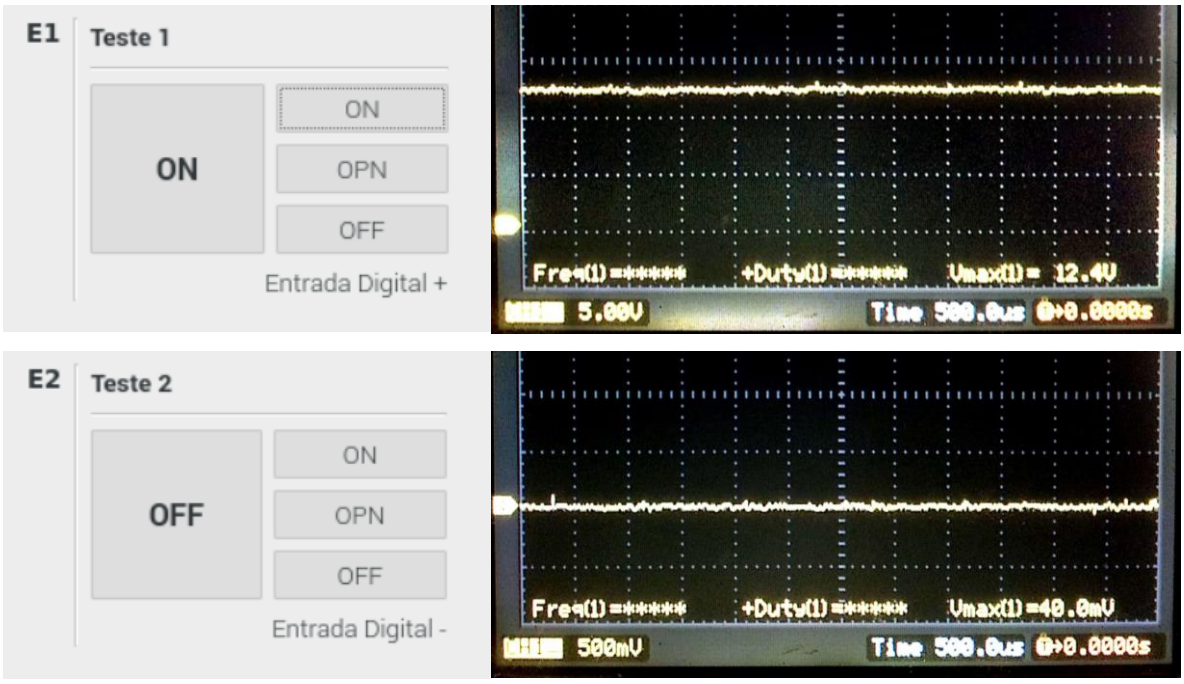
Figura 77 – Ondulação saída analógica



Fonte: Elaborado pelo autor.

Por fim, as saídas digitais realizam somente o controle *ON/OFF* do sistema. Os resultados são exibidos na Figura 78. O estado aberto comanda o relé para a saída analógica mantendo a saída flutuante, não sendo apresentado abaixo.

Figura 78 – Resultado das Saídas digitais

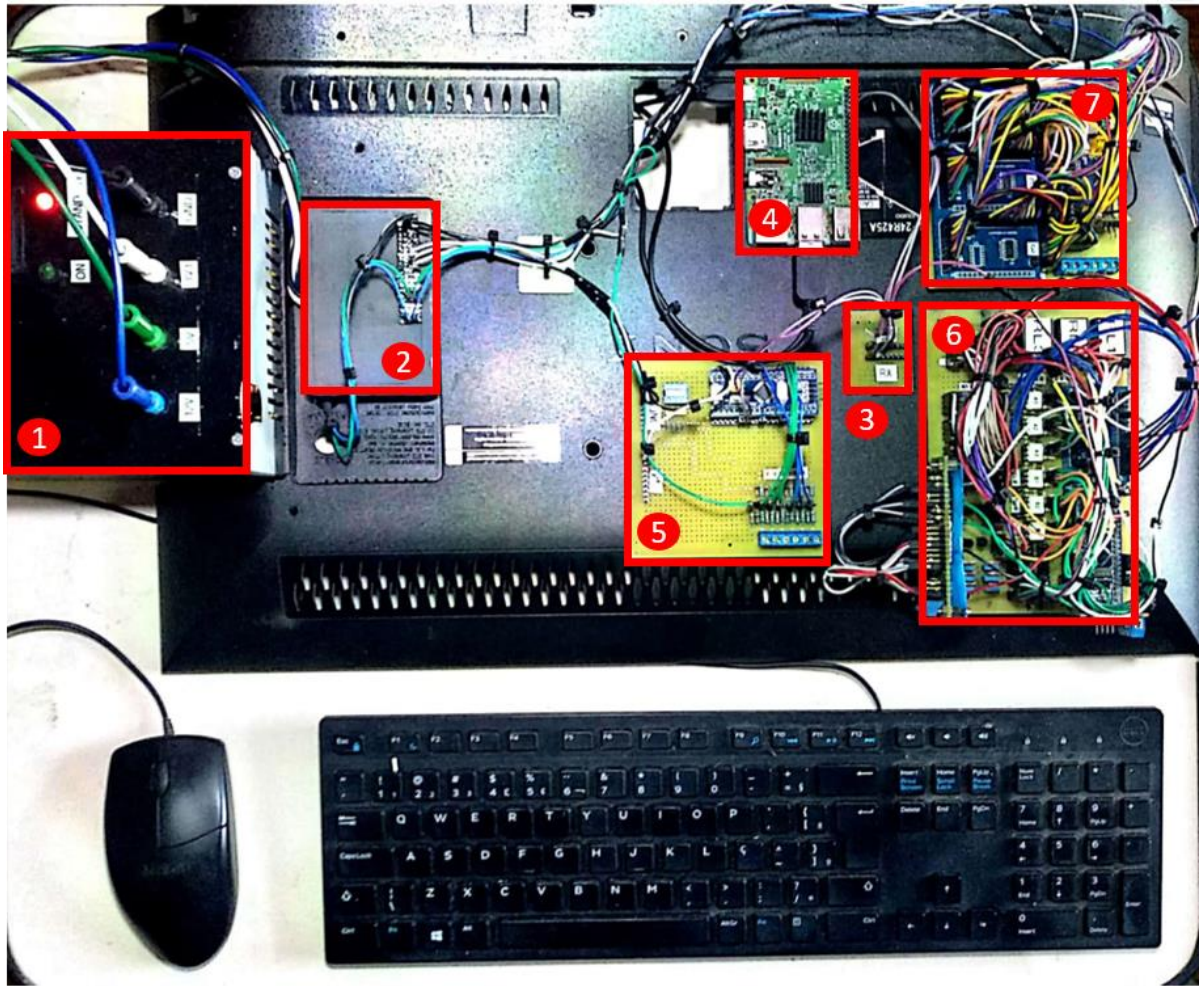


Fonte: Elaborado pelo autor.

4.4 Validação

A Figura 79 apresenta o *setup* completo de teste. Os itens estão identificados conforme a numeração: 1 – fonte de alimentação utilizada; 2 – barramento de alimentação; 3 – barramento serial; 4 – Raspberry Pi; 5 – módulo de entradas; 6 – módulo de saídas e 7 – módulo de resistências. Para a validação final, foi utilizado um produto da empresa, que apresenta uma entrada de pressostato digital, três saídas digitais de ventilação, uma saída digital para o compressor e dois sensores de temperatura, permitindo o teste com somente um módulo de cada.

Figura 79 – Setup completo



Fonte: Elaborado pelo autor.

4.5 Custos envolvidos

Se comparado a sistemas comerciais existentes, o protótipo desenvolvido possui um custo significativamente inferior, visto que foi construído para aplicações que não exijam tanta exatidão nas medições.

Assim, os custos envolvidos para a confecção do protótipo são apresentados no **Erro! Fonte de referência não encontrada.**. Alguns itens foram importados e, desta forma, seus valores foram convertidos na cotação do dia 22 de outubro de 2020, na qual a equivalência era de US$ 1,00 = R$ 5,58.

Tabela 7 – Custos protótipo

Item	Qnt	Unidade	Custo unitário (R$)	Custo total (R$)
Raspberry	1	unid	379,90	379,90
Fonte de alimentação ATX	1	unid	63,90	63,90
Cabo rígido 22 AWG	2	m	0,71	1,42
Terminais banana	4	unid	1,42	5,68
Borne para cabo banana	4	unid	1,19	4,76
Chave gangorra 2 terminais	1	unid	1,04	1,04
Placas de prototipação	3	unid	5,00	15,00
Bluepill STM32F103C8T6	3	unid	32,90	98,70
Potenciômetro AD5241	5	unid	15,79	78,96
Potenciômetro AD175	5	unid	22,77	113,85
MOSFET IRF7343	3	unid	5,19	15,57
LM358	3	unid	0,94	2,82
Relé 12V	3	unid	2,61	7,83
Diodo 1N4148	15	unid	0,14	2,10
BC327	5	unid	0,21	1,05
BC337	10	unid	0,19	1,90
PCB Adaptadora SMD/PTH JLCPCB	5	unid	1,12	5,60
			Somatório	**800,08**

Fonte: Elaborado pelo autor.

5 CONCLUSÃO

A proposta central do presente projeto era a aplicação e ampliação de conhecimentos adquiridos ao longo do curso de graduação em Engenharia Elétrica, aliada a uma limitação identificada na empresa alvo referente ao desenvolvimento de novos módulos de teste específicos para cada novo produto, trabalho que resulta em um grande volume de equipamentos de teste.

Frente à essa problemática, o protótipo em questão, atendendo as especificações propostas inicialmente, possibilitou o teste de controladores automotivos de modo genérico. As leituras de entradas, acionamento de saídas e emulação de sensores de temperatura funcionaram corretamente, apresentando resultados satisfatórios, mas ainda com questões a serem aprimoradas.

Aponta-se que existem diversas oportunidades de possíveis aperfeiçoamentos, com a intenção de alcançar um melhor desempenho por parte dos módulos em conjunto com a IHM. Tais oportunidades se tornaram evidentes ao decorrer do desenvolvimento do projeto, tanto em termos de *hardware*, quanto em termos de *software*.

A IHM em questão poderia apresentar novas funcionalidades dedicadas também a testes que ingressam em modalidades fora do teste de aceitação, a exemplo do teste de funções software, servindo como ferramenta e auxílio. Para tanto, poderiam ser desenvolvidos *log* de dados referentes a cada ação de controle efetuada na interface, ou mesmo um espaço para anotação de observações. Outra opção disponível seria se aprofundar ainda mais no teste de aceitação, criando roteiros de teste dentro da própria IHM e, assim, criando um banco de dados para os controladores.

Além disso, em um segundo momento, já com o embasamento desta primeira implementação, poderiam ser estudados outros microcontroladores dedicados a cada módulo, de modo a minimizar o custo e maximizar o desempenho. O módulo de resistências utilizando potenciômetros digitais, por exemplo, poderia conter um bloco de realimentação utilizando divisores de tensão com resistores de alta precisão em conjunto com um conversor A/D de alta resolução, de forma a possibilitar a medição exata da resistência ajustada pelos potenciômetros. O funcionamento em conjunto com a IHM para cálculo da temperatura equivalente conforme parâmetros de entrada também é uma futura implementação bastante interessante para a aplicação.

Há, também, a possibilidade de confecção de novos módulos que, por se tratar de um protótipo inicial, não foram incluídos por exigirem um estudo e maior preparação por parte da empresa. O módulo de comunicações, mencionado no início do trabalho, juntamente a um módulo que contempla um teste de válvulas *HVAC* (*Heating Ventilation and Air* Conditioning, ou, do português, Aquecimento, Ventilação e Ar-condicionado) automotivas seriam um complemento significativo para a validação de controladores automotivos.

Finalmente, referente ao custo envolvido no projeto, em uma perspectiva geral, pode ser considerado mínimo para investimento de uma empresa, visto que elimina a necessidade da confecção de novos módulos para cada novo produto. Exemplificando através da empresa alvo, estima-se que a confecção de cada novo módulo leve em torno de cinco horas para confecção, com uma média de desenvolvimento de um novo módulo a cada dois dias. É possível perceber que, facilmente, o investimento em um protótipo de módulos de teste generalizado se compensaria em pouco tempo.

REFERÊNCIAS

AGTIC UFPR. **Técnicas de Teste**. Disponivel em: <https://www.agtic.ufpr.br/pds-ufpr/ProcessoDemoiselllePlugin/guidances/supportingmaterials/tecnicasTestes_8AB32ED1.html>. Acesso em: 2 Dezembro 2019.

ALEXANDER, C. K.; SADIKU, M. Fundamentos de Circuitos Elétricos. In: ALEXANDER, C. K.; SADIKU, M. **Resposta de frequência**. 5ª. ed. [S.l.]: AMGH, 2013. p. 569-570.

ANALOG DEVICES. Analog Devices. **AD5175**, 2010. Disponivel em: <https://www.analog.com/media/en/technical-documentation/data-sheets/AD5175.pdf>. Acesso em: 8 Março 2020.

ANALOG DEVICES. Analog Devices. **AD5241**, 2015. Disponivel em: <https://www.analog.com/media/en/technical-documentation/data-sheets/AD5241_5242.pdf>. Acesso em: 8 Março 2020.

ARDUÍNO. Arduino Introduction, 13 Outubro 2019. Disponivel em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 12 Outubro 2019.

ARM LIMITED. **ARM Architecture**, 2005. Disponivel em: <https://www.scss.tcd.ie/~waldroj/3d1/arm_arm.pdf>. Acesso em: 11 Setembro 2019.

ARM LIMITED. **Programmer's Guide for ARMv8-A**, 24 Março 2015. Disponivel em: <https://static.docs.arm.com/den0024/a/DEN0024A_v8_architecture_PG.pdf?_ga=2.264447244.204749681.1568330492-586459567.1567979175>. Acesso em: 20 Setembro 2019.

ARORA, R. **I2C Bus Pullup Resistor Calculation**, Fevereiro 2015. Disponivel em: <http://www.ti.com/lit/an/slva689/slva689.pdf>. Acesso em: 21 Setembro 2019.

ASWINTH, R. **Getting Started with STM32 using Arduino IDE:** Blinking LED, 8 Agosto 2018. Disponivel em: <https://circuitdigest.com/microcontroller-projects/getting-started-with-stm32-development-board-stm32f103c8-using-arduino-ide>. Acesso em: 18 Outubro 2019.

BEAGLEBOARD ORG. **What is BeagleBone Black?**, 26 Setembro 2019. Disponivel em: <https://beagleboard.org/black>. Acesso em: 16 Novembro 2019.

BILL, G. **IoT Microcontrollers Have ADCs, but Know When to Choose and Apply an External ADC**, 21 nov. 2018. Disponivel em: <https://www.digikey.com/en/articles/techzone/2018/nov/iot-microcontrollers-have-adcs-know-when-apply-an-external-adc>. Acesso em: 22 Novembro 2019.

BOSCH. **Electric Motors**, 2019. Disponivel em: <https://www.bosch-ibusiness.com/media/images/products/dc_motors/xx_pdfs_2/electric_motors_catalogue_en_2019-2020_low.pdf>. Acesso em: 22 Novembro 2019.

CONTEC COMPANY. **Digital I/O Basic Knowledge**, 2020. Disponivel em: <https://www.contec.com/support/basic-knowledge/daq-control/digital-io/>. Acesso em: 15 Junho 2020.

CURTO CIRCUITO. **Placa ARM STM32 - STM32F103C8T6**, 2019. Disponivel em: <https://www.curtocircuito.com.br/placa-arm-stm32-stm32f103c8t6.html>. Acesso em: 14 Novembro 2019.

DIGI-KEY ELECTRONICS. **Protecting Inputs in Digital Electronics**, 04 Novembro 2012. Disponivel em: <https://www.digikey.com/en/articles/techzone/2012/apr/protecting-inputs-in-digital-electronics>. Acesso em: 16 Novembro 2019.

D-ROBOTICS UK. **DHT11 Humidity & Temperature Sensor**, 2010. Disponivel em: <https://datasheetspdf.com/pdf-file/785590/D-Robotics/DHT11/1>. Acesso em: 9 Agosto 2020.

ELECTRONICS SOLUTION. **ONEPASS:** Innovative Test System, 2019. Disponivel em: <http://www.electronicsolution.it/onepass-il-sistema-di-test-innovativo/?lang=en>. Acesso em: 2 Dezembro 2019.

EMBEDDED STAFF. **Serial Protocols Compared**, 2002. Disponivel em: <https://www.embedded.com/serial-protocols-compared/>. Acesso em: 9 Agosto 2020.

FEUP. FEUP - Universidade de Porto. **Tabela ASCII**, 2003. Disponivel em: <https://web.fe.up.pt/~ee96100/projecto/Tabela%20ascii.htm>. Acesso em: 15 Março 2020.

FLUKE CORPORATION. **What is a digital multimeter?**, 2019. Disponivel em: <https://www.fluke.com/en-us/learn/best-practices/measurement-basics/electricity/what-is-a-digital-multimeter>. Acesso em: 22 Novembro 2019.

FRUETT, F. Unicamp. **Eletrônica básica I**, 2013. Disponivel em: <http://www.dsif.fee.unicamp.br/~fabiano/EE530/PDF/C%20-%20Circuitos%20com%20operacionais.pdf>. Acesso em: 27 Junho 2020.

GORDON PROJECTS. Drogon.projects. **Serial Library**, 2012. Disponivel em: <https://projects.drogon.net/raspberry-pi/wiringpi/serial-library/>. Acesso em: 06 mar. 2020.

HBM. **O que é condicionador de sinal?**, 2019. Disponivel em: <https://www.hbm.com/pt/7339/o-que-e-um-condicionador-de-sinal-funcoes/>. Acesso em: 12 Outubro 2019.

IMPAC INSTRUMENTOS DE MEDIÇÃO. **Módulo Entrada Digital**, 2020. Disponivel em: <https://www.impac.com.br/modulo-entrada-digital/modulo-entrada-digital-isolada-14-entradas-saida-rs-485-ip-7041.html>. Acesso em: 9 Agosto 2020.

INFINEON TECHNOLOGIES. **IRF7343PbF**, 2020. Disponivel em: <https://www.infineon.com/dgdl/irf7343pbf.pdf?fileId=5546d462533600a4015355f68c1a1b73>. Acesso em: 6 Junho 2020.

KEYSIGHT TECHNOLOGIES. **TS-3000 Smart Meter Functional Test System**, 2019. Disponivel em: <https://www.keysight.com/pt/pd-2039128-pn-E2242A/smart-meter-functional-test-system>. Acesso em: 2 Dezembro 2019.

KEYSIGHT TECHNOLOGIES. **TS-5040 Functional Test System**, 2019. Disponivel em: <https://www.keysight.com/pt/pd-1264863-pn-E2230B/functional-test-system>. Acesso em: 2 Dezembro 2019.

KEYSIGHT TECHNOLOGIES. **Digital Multimeters (DMM)**, 2020. Disponivel em: <https://www.keysight.com/en/pcx-2832403/digital-multimeters-dmm?cc=BR&lc=por>. Acesso em: 09 Setembro 2020.

LAZAR, G.; PENEA, R. Mastering QT. In: LAZAR, G.; PENEA, R. **Mastering QT**. Birmingham: Packt Publishing Ltd., 2016. p. Prefácio.

LEENS, F. **An introduction to I2C and SPI protocols**, p. 8-13, 2009. Disponivel em: <https://ieeexplore.ieee.org/document/4762946>. Acesso em: 21 Setembro 2019.

LEI-GAO, W.; CHUN-PING, W.; MING, L. **Design of Digital Multimeter Module Based on ARM**, 2010. Disponivel em: <https://ieeexplore.ieee.org/document/5610193>. Acesso em: 16 Novembro 2019.

LITTELFUSE. **PolySwitch® Resettable Devices**, 2020. Disponivel em: <https://www.littelfuse.com/products/polyswitch-resettable-ptcs.aspx>. Acesso em: 16 Outubro 2020.

MANKAR, J. et al. **Review of I2C Protocol**, Janeiro 2014. Disponivel em: <http://www.ijrat.org/downloads/Vol-2/jan-2014/paper%20ID-21201448>. Acesso em: 21 Setembro 2019.

MARVINTEST. **ATEasy**, 2020. Disponivel em: <https://www.marvintest.com/Product.aspx?model=ATEasy&Tab=Whats+New?>. Acesso em: 9 Agosto 2020.

MATLOFF, N. University of California at Davis. **Cyclic Redudancy Check**, 2001. Disponivel em: <Cyclic Redundancy Checking>. Acesso em: 9 Agosto 2020.

MAXIM INTEGRATED PRODUCTS. **MAX5487/MAX5488/MAX5489**, 2010. Disponivel em: <https://datasheets.maximintegrated.com/en/ds/MAX5487-MAX5489.pdf>. Acesso em: 23 Outubro 2019.

MILLER, R. W.; COLLINS, C. T. **Acceptance Testing**, 2002. Disponivel em: <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/recursos/Testing05.pdf>. Acesso em: 2 Dezembro 2019.

MYERS, P. **Interfacing using Serial Protocols SPI and I2C**, 2007. Disponivel em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.460.2531&rep=rep1&type=pdf>. Acesso em: 21 Setembro 2019.

NATIONAL INSTRUMENTS. **Choosing Between Line Driver, Open Collector, and Push Pull Encoders for NI Device**, 2019. Disponivel em: <https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019MXOSA2&l=pt-BR>. Acesso em: 9 Agosto 2020.

NATIONAL INSTRUMENTS. **Conceitos básicos do ambiente LabVIEW**, 2019. Disponivel em: <https://www.ni.com/getting-started/labview-basics/pt/environment>. Acesso em: 2 Novembro 2019.

NATIONAL INSTRUMENTS. **Introduction to Data Acquisition**, 14 Março 2019. Disponivel em: <https://www.ni.com/pt-br/innovations/white-papers/06/introduction-to-data-acquisition.html>. Acesso em: 19 Outubro 2019.

NATIONAL INSTRUMENTS. **O que é aquisição de dados?**, 2019. Disponivel em: <http://www.ni.com/data-acquisition/what-is/pt/>. Acesso em: 12 Outubro 2019.

NAYYAR, A.; PURI, V. IEEE. **A review of Arduino board's, Lilypad's & Arduino shields**, 16 - 18 Março 2016. Disponivel em: <https://ieeexplore.ieee.org/document/7724514>. Acesso em: 12 Outubro 2019.

NEXPERIA. Nexperia. **74HC4053 - Triple 2-channel analog multiplexer/demultiplexer**, 2020. Disponivel em: <https://assets.nexperia.com/documents/data-sheet/74HC_HCT4053.pdf>. Acesso em: 2 Março 2020.

ON SEMICONDUCTOR. **Everything You Wanted to Know About Digital Potentiometer (POT)**, 2013. Disponivel em: <https://www.onsemi.com/pub/Collateral/AND8414-D.PDF>. Acesso em: 19 Outubro 2019.

OTTE, K. **Insights into Switching Technology**, 25 Agosto 2014. Disponivel em: <https://blog.pickeringtest.com/what-is-a-programmable-resistor>. Acesso em: 18 Outubro 2019.

PEREIRA, F. Tecnologia ARM : microcontroladores de 32 BITS. In: PEREIRA, F. **Tecnologia ARM:** microcontroladores de 32 BITS. 1ª. ed. São Paulo: Erica, 2007. Cap. 1, p. 17, 22, 66.

PICKERING INTERFACES. **PXI 2.5W Programmable Resistor Module, 2-Channel, 2.5Ω to 1.51MΩ**, 2020. Disponivel em: <https://www.pickeringtest.com/en-br/product/40-251-044-pxi-2-5w-programmable-resistor-2-channel-2-5r-to-1-51m>. Acesso em: 09 Setembro 2020.

PICKERING TEST INTERFACES. **PXI High Density Precision Resistor Module**, 2019. Disponivel em: <https://www.pickeringtest.com/content/downloads/datasheets/40-297D.pdf>. Acesso em: 18 Outubro 2019.

PICKERING TEST INTERFACES. Pickering. **About us**. Disponivel em: <https://www.pickeringtest.com/about-us>. Acesso em: 15 Setembro 2019.

QT GROUP. **Why QT?** Disponivel em: <https://www.qt.io>. Acesso em: 3 Novembro 2019.

QT GROUP. **Qt Quick Controls - Imagine Style Example:** Automotive, 2019. Disponivel em: <https://doc-snapshots.qt.io/qt5-dev/qtquickcontrols2-examples.html>. Acesso em: 3 Novembro 2019.

QT GROUP. **Qt Quick Controls - Imagine Style Example:** Music Player, 2019. Disponivel em: <https://doc-snapshots.qt.io/qt5-dev/qtquickcontrols-imagine-musicplayer-example.html#>. Acesso em: 3 Novembro 2019.

RASPBERRY PI ORG. **Raspberry Pi 4 Model B**, 2019. Disponivel em: <https://www.raspberrypi.org/products/>. Acesso em: 16 Novembro 2019.

RASPBERRY PI ORG. **UART Configuration**, 2020. Disponivel em: <https://www.raspberrypi.org/documentation/configuration/uart.md>.

RICHARDSON, M.; WALLACE, S. **Getting Started with Raspberry Pi**. Sebastopol: O'Reilly Media, 2013.

RIEDON RESISTOR MANUFACTURING. **Understanding Resistors and Temperature**, 2014. Disponivel em: <https://riedon.com/media/pdf-tech/Understanding_Resistors_and_Temperature.pdf>. Acesso em: 15 Agosto 2020.

RYZHYK, L. **The ARM Architecture**, 5 Junho 2006. Disponivel em: <https://www.researchgate.net/profile/Leonid_Ryzhyk/publication/228392292_The_ARM_Architecture/links/554b73c40cf29f836c96b2b7.pdf>. Acesso em: 10 Setembro 2019.

SET GMBH. **ATS 420X**, 2019. Disponivel em: <https://www.smart-e-tech.de/en/hil-functional-test-systems/functional-test-systems/>. Acesso em: 2 Dezembro 2019.

SOUZA, D. J. D. **Desbravando o PIC:** ampliado e atualizado para PIC 16F628A. 12ª. ed. São Paulo: Erica, 2009. 21 - 23 p.

STAFF, M. I. **Powerful Low-Cost Arduino Alternatives:** STM32 Nucleo, 20 Junho 2016. Disponivel em: <https://www.digikey.com/en/maker/blogs/st-nucleo-a-powerful-low-cost-alternative-to-the-arduino>. Acesso em: 18 Outubro 2019.

STEFFES, M. **RLC Filter Design for ADC Interface Applications**, Janeiro 2005. Disponivel em: <http://www.ti.com/lit/an/sbaa108a/sbaa108a.pdf>. Acesso em: 22 Novembro 2019.

STMICROELETRONICS. **Datasheet STM32F103x8/STM32F103xB**, 2015. Disponivel em: <https://www.st.com/resource/en/datasheet/stm32f103c8.pdf>. Acesso em: 22 Setembro 2019.

STMICROELETRONICS. **Reference Manual STM32F103C8T6**, 2018. ISSN RM0008 Rev 20. Disponivel em: <https://www.st.com/resource/en/reference_manual/cd00171190-stm32f101xx-stm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf>. Acesso em: 15 Março 2020.

TECHOPEDIA. Techopedia. **Cyclic Redundancy Check (CRC)**, 2016. Disponivel em: <https://www.techopedia.com/definition/1793/cyclic-redundancy-check-crc>. Acesso em: 9 Agosto 2020.

TEXAS INSTRUMENTS. **BeagleBone Black Development Board**, 2014. Disponivel em: <http://www.ti.com/tool/BEAGLEBK#descriptionArea>. Acesso em: 16 Novembro 2019.

TEXAS INSTRUMENTS. **BeagleBone Black Development Board**, 07 Agosto 2019. Disponivel em: <http://www.ti.com/lit/wp/spry235a/spry235a.pdf>. Acesso em: 16 Novembro 2019.

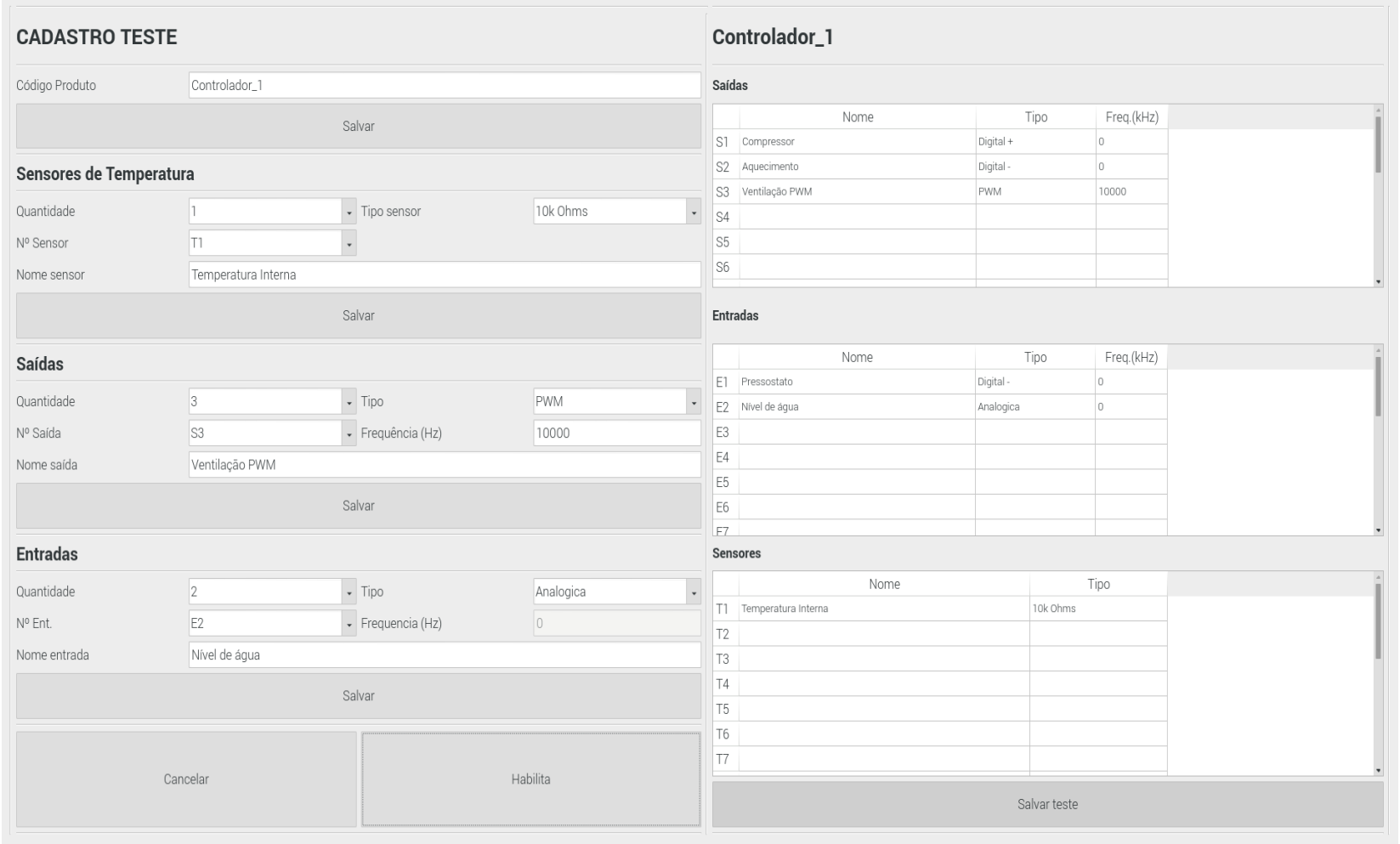
TRAVIS, J.; KRING, J. **Labview for everyone**. 3ª. ed. Crawfordsville: Pearson Education, 2006.

VALDEZ, J.; BECKER, J. **Understanding the I2C**, Junho 2015. Disponivel em: <http://www.ti.com/lit/an/slva704/slva704.pdf>. Acesso em: 18 Outubro 2019.

VISHAY. **Small Signal Fast Switching Diodes 1N4148**, 2019. Disponivel em: <https://www.vishay.com/docs/81857/1n4148.pdf>. Acesso em: 19 Março 2020.

APÊNDICE A – TELA DE CADASTRO DE NOVO TESTE

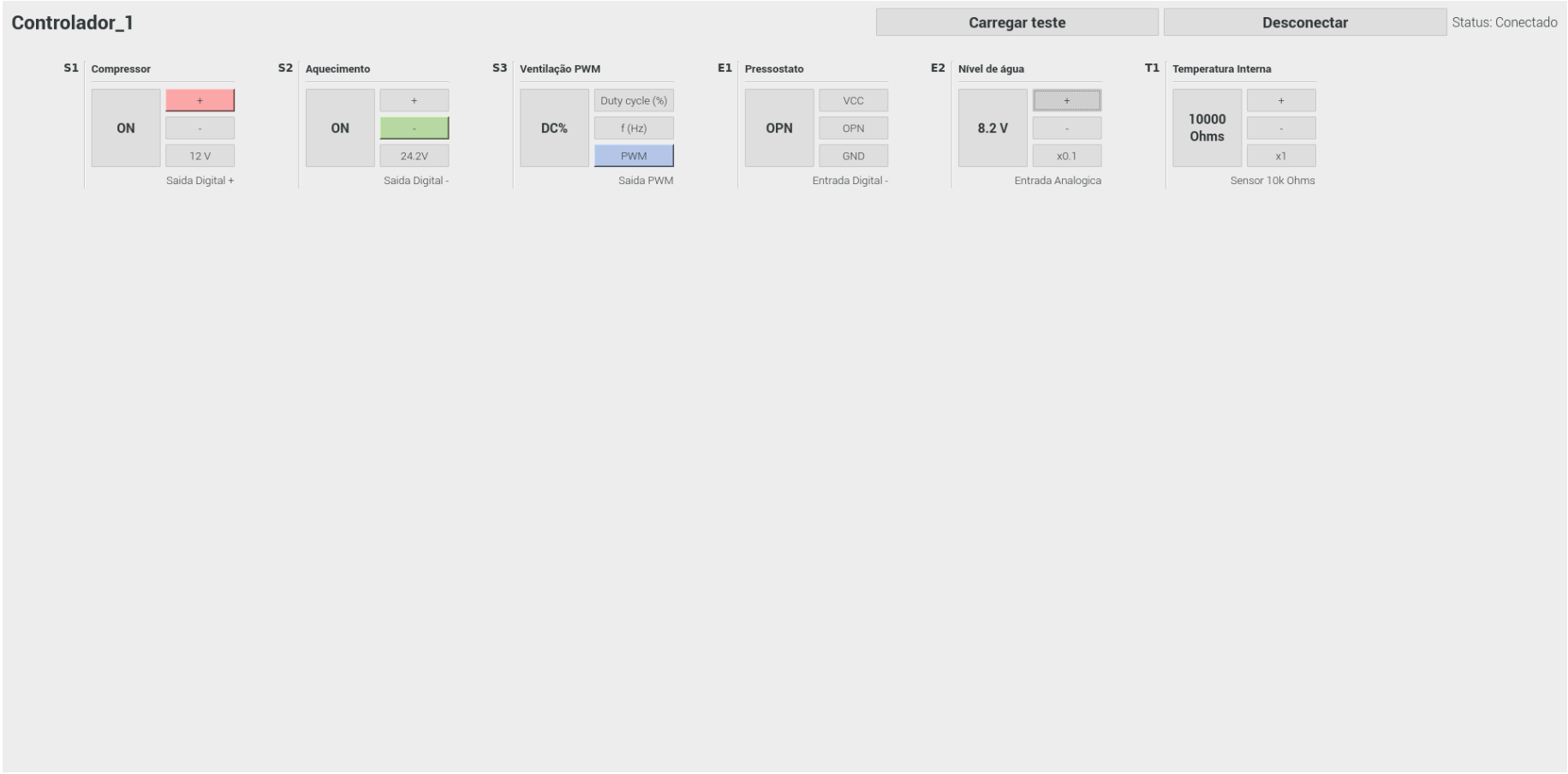
Figura 80 – Tela de cadastro de teste completa



Fonte: Elaborado pelo autor.

APÊNDICE B – TELA DE TESTE CARREGADA

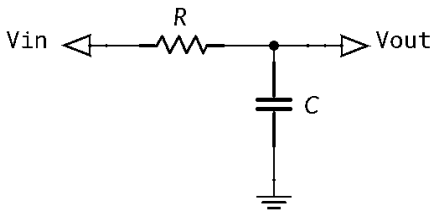
Figura 81 – Tela de teste completa



Fonte: Elaborado pelo autor.

APÊNDICE C – CÁLCULO DE FILTRO PASSA-BAIXAS

Figura 82 – Filtro passa-baixas passivo



Fonte: Elaborado pelo autor.

Encontramos a função de transferência de V_{OUT} em relação a V_{IN}, utilizando o domínio da frequência e o método de nós para obtenção das equações. O resultado é apresentado na Equação (7) (ALEXANDER e SADIKU, 2013):

$$\frac{V_{OUT} - V_{IN}}{R} + V_{OUT}sC = 0 \tag{4}$$

$$V_{IN} = V_{OUT} + V_{OUT}sRC \tag{5}$$

$$V_{IN} = V_{OUT}(1 + sRC) \tag{6}$$

$$\frac{V_{OUT}}{V_{IN}} = \frac{1}{sRC + 1} \tag{7}$$

Na prática, se desejarmos usar mais um filtro RC em série com este calculado, efetivamente acrescentaríamos mais um polo no filtro, e o decaimento do ganho na frequência de corte é duplicado. Entretanto, para fins de cálculo, podemos manter apenas com um circuito RC. (ALEXANDER e SADIKU, 2013)

Sabemos que:

$$G(s) = \frac{V_{OUT}}{V_{IN}} \tag{8}$$

Onde:

$G(s)$ é função de transferência;

Então, temos que:

$$G(s) = \frac{1}{sRC + 1} \tag{9}$$

A frequência de corte é obtida quando a amplitude da função de transferência é igual a:

$$\frac{1}{\sqrt{2}} = \left|\frac{V_{OUT}}{V_{IN}}\right| \quad (10)$$

Logo, convertendo s para $j\omega$:

$$\frac{1}{\sqrt{2}} = \frac{1}{\sqrt{\omega^2 R^2 C^2 + 1}} \quad (11)$$

$$\sqrt{\omega^2 R^2 C^2 + 1} = \sqrt{2} \quad (12)$$

$$\omega^2 R^2 C^2 + 1 = 2 \quad (13)$$

$$\omega^2 = \frac{1}{R^2 C^2} \quad (14)$$

$$\omega = \frac{1}{RC} \quad (15)$$

Onde:

ω é a frequência de corte na forma angular;

$$f = \frac{\omega}{2\pi} \quad (16)$$

$$f = \frac{1}{2\pi RC} \quad (17)$$

Como desejamos uma frequência de corte de 2 kHz, podemos escolher um valor comercial para R, como 1 kΩ e encontrar o capacitor:

$$2000 = \frac{1}{2\pi 1000 C}$$

$$C = \frac{1}{2\pi 1000 * 2000}$$

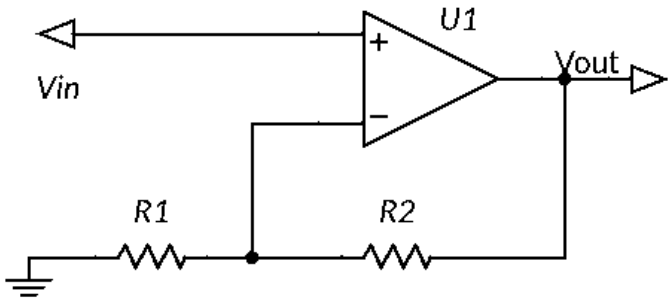
$$C \cong 80\, nF$$

Para um valor comercial do capacitor, utilizaremos 100 nF.

$$C = 100\, nF$$

APÊNDICE D – CÁLCULO AMPLIFICADOR NÃO-INVERSOR

Figura 83 – Amplificador não-inversor



Fonte: Elaborado pelo autor.

É sabido que a equação do amplificador não-inversor é dada pela Equação (18) (FRUETT, 2013):

$$\frac{Vout}{Vin} = \frac{R2}{R1} + 1 \tag{18}$$

Considerando que desejamos um ganho de aproximadamente 1, foi utilizado $R1 = 10\ k\Omega$ e $R2 = 1\ k\Omega$, de modo a obter:

$$\frac{Vout}{Vin} = \frac{1000}{10000} + 1 \tag{19}$$

$$\frac{Vout}{Vin} = 0{,}1 + 1$$

$$\frac{Vout}{Vin} = 1{,}1$$