

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE
SANTA CATARINA - CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE ENGENHARIA ELETRÔNICA**

JOÃO VITOR BASSANI

**DESENVOLVIMENTO DE UM KIT DE ROBÓTICA DE
BAIXO CUSTO APLICADO AO ENSINO NO IFSC**

Florianópolis, 2020

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE
SANTA CATARINA - CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE ENGENHARIA ELETRÔNICA**

JOÃO VITOR BASSANI

DESENVOLVIMENTO DE UM KIT DE ROBÓTICA DE BAIXO CUSTO APLICADO AO ENSINO NO IFSC

Trabalho de conclusão de curso submetido
ao Instituto Federal de Educação, Ciência
e Tecnologia de Santa Catarina como parte
dos requisitos para obtenção do título de
engenheiro eletrônico

Orientador:
Prof. Dr. Joabel Moia

Florianópolis, 2020

Ficha de identificação da obra elaborada pelo autor.

Bassani, João Vitor

Desenvolvimento de um kit de robótica de baixo custo aplicado ao ensino no IFSC / João Vitor Bassani ; orientação de Joabel Moia. - Florianópolis, SC, 2020.
150 p.

Trabalho de Conclusão de Curso (TCC) - Instituto Federal de Santa Catarina, Câmpus Florianópolis. Bacharelado em Engenharia Eletrônica. Departamento Acadêmico de Eletrônica.
Inclui Referências.

1. Robótica. 2. Kit Didático. 3. Robótica Pedagógica Livre. I. Moia, Joabel. II. Instituto Federal de Santa Catarina. Departamento Acadêmico de Eletrônica. III. Título.

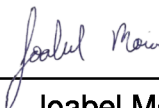
DESENVOLVIMENTO DE UM KIT DE ROBÓTICA DE BAIXO CUSTO APLICADO AO ENSINO NO IFSC

JOÃO VITOR BASSANI

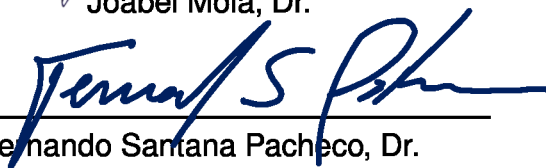
Este Trabalho foi julgado adequado para obtenção do Título de Engenheiro Eletrônico em novembro de 2020 e aprovado na sua forma final pela banca examinadora do Curso de Engenharia Eletrônica do Instituto Federal de Educação Ciência, e Tecnologia de Santa Catarina.

Florianópolis, 04 de novembro, 2020.

Banca Examinadora:



Joabel Moia, Dr.



Fernando Santana Pacheco, Dr.



Renan Augusto Starke, Dr.

AGRADECIMENTOS

Aos meus pais, Moises e Susineia, agradeço por me permitirem alcançar novos horizontes e dar suporte incondicional às minhas vontades de explorar o mundo. Não haveria esforço e muito menos êxito em minhas tentativas se não me fosse dada a oportunidade de viver uma nova realidade, repleta de apoio e incentivo na jornada em busca de minhas realizações.

Ao meu orientador, Joabel Moia, agradeço pelo estímulo para abordar uma temática tão rica, vasta e necessária, imprimindo um significado ao projeto capaz de alimentar ainda mais a motivação em executá-lo. Também, reforço o agradecimento pelo suporte dado para ultrapassar todos os momentos difíceis e os percalços inimagináveis que permearam o desenvolvimento deste projeto.

À instituição e demais professores, agradeço pela oportunidade dada de construir uma base sólida para as próximas etapas de minha vida e minha carreira, me alimentando com confiança e conhecimento. Sem as instituições de ensino, nosso fracasso como sociedade já teria sido alcançado, por isso, devem ser defendidas a qualquer custo.

*"Educar verdadeiramente não é ensinar fatos
novos ou enumerar fórmulas prontas, mas sim
preparar a mente para pensar."
(Albert Einstein)*

RESUMO

Dialogar sobre o futuro da tecnologia e seu impacto no desenvolvimento educacional é um debate urgente e necessário, que envolve fatores políticos, econômicos e sociais complexos. Enquanto o mundo tende a uma realidade completamente imersa em soluções tecnológicas para problemas cotidianos, as instituições de ensino brasileiras veem-se rodeadas por condições inviáveis para execução dos seus planos de ensino e para o desenvolvimento humano básico que é encontrado em ambientes pedagógicos. Toda essa situação se constroi como reflexo da escassez de recursos e de uma construção social de grandes disparidades socioeconômicas e barreiras culturais, capazes de atrapalhar o ingresso e a permanência dos estudantes aptos para o ensino, resultando principalmente na marginalização destes indivíduos.

Neste estudo, numa busca pelo entendimento do panorama atual do ensino superior e das condições que direcionam os estudantes a esses ambientes, o foco principal é traçar uma alternativa viável capaz de interferir nessas realidades e sugerir uma solução simples e acessível que permita a aproximação do ensino de robótica a todos os cenários e níveis de aprendizado, procurando por recursos que tenham viabilidade em aspecto construtivo e econômico.

A ideia geral para o projeto apresentado é proporcionar um documento de referência que descreva a construção de um *kit* de iniciação ao estudo da robótica, focando na acessibilidade às informações disponibilizadas e em um alto índice de reprodutibilidade, contendo componentes fáceis de acessar, mesmo em um país com poucos recursos tecnológicos, e dependendo apenas do uso de ferramentas gratuitas de desenvolvimento. Ao longo dos capítulos, o trabalho descreve a construção de uma plataforma personalizada de *hardware*, acompanhada de um programa completo, embarcado em seu núcleo microcontrolado, que permite o gerenciamento das funções principais estabelecidas ao longo da pesquisa e estudo inicial.

Todo o desenvolvimento do texto é focado na apresentação de cada detalhe de decisão sobre os aspectos construtivos do conjunto, permitindo que em ações posteriores possa-se entender a progressão das alterações realizadas, ou mesmo incentivar a personalização de todos estes pontos, garantindo que além de poder reproduzir nos moldes propostos, o usuário possa inspirar-se a criar uma solução que se adapte à sua própria realidade.

Palavras-chave: Robótica Educacional. *Kit* Didático. Robótica Pedagógica Livre.

ABSTRACT

The discussion surrounding the future of technology and all the impact caused on educational development is not just a necessary but also an urgent debate, involving a series of political, economic and social factors. While the world follows a trend towards a reality completely immersed in technological solutions for everyday problems, Brazilian educational institutions find themselves surrounded by unsustainable conditions to execute even the most basic teaching agenda over the human development that should be found in pedagogical environments, reflecting a lack of resources and a social construction composed by huge socioeconomic disparities and cultural barriers, hindering the entry and permanence of students in academic environments, resulting mainly in the marginalization of these individuals.

In this research, looking for a clear view over the current panorama in college education and the conditions that leads students towards these environments, the focus is to determine a conceivable alternative to interfere in these situations and suggest accessible solutions to include some robotics to the teaching process, including all scenarios and education levels, defining practical and realistic resources, counting on both executive and economic aspects.

The main goal turns to the conception of a reference document that could describe the construction of a kit dedicated to basic studies about robotics, focusing on accessibility to the available information and a high reproducibility condition, with components that are easily found even in a country with few technological resources and depending exclusively on the use of free development tools. Through some chapters, this work describes the construction of a customized hardware platform, working along a complete program embedded in its microcontrolled core, responsible for the management of the main functions established throughout the research and initial study.

The entire development of this text is based on the presentation on every detail of every decision made, following the constructive aspects of the set and allowing later understanding of the whole process, giving the basic necessary comprehension of the executed changes, even encouraging the customization of all these points, ensuring that in addition to being able to reproduce the system, a user can be inspired to create a solution adapted to his own reality.

Keywords: *Educational Robotics. Educational kit. Free Educational Robotics.*

LISTA DE ILUSTRAÇÕES

Figura 1 – Bloco configurável da primeira geração do <i>LEGO Mindstorms®</i> . . .	28
Figura 2 – Blocos configuráveis da segunda (esq.) e terceira (dir.) gerações do <i>LEGO Mindstorms®</i>	29
Figura 3 – Bloco configurável do <i>LEGO Mindstorms®EV3</i> e seus módulos adicionais	30
Figura 4 – Kit de desenvolvimento TI-RSLK®	32
Figura 5 – Módulos associados ao kit de desenvolvimento TI-RSLK®	32
Figura 6 – As soluções <i>mTiny</i> (esq.), <i>Neuron</i> (centro) e <i>Codey Rocky</i> (dir.) . . .	34
Figura 7 – Placa de desenvolvimento <i>HaloCode</i>	35
Figura 8 – Modelos <i>mBot</i> (esq.), <i>mBot Ranger</i> (centro) e <i>mBot Ultimate 2.0</i> (dir.)	36
Figura 9 – Modelos <i>Sphero® BOLT</i> (esq.)e <i>RVR</i> (dir.)	38
Figura 10 – Placas de desenvolvimento <i>Arduino Nano®</i> (esq.) e <i>Arduino Due®</i> (dir.)	42
Figura 11 – Placa de desenvolvimento do modelo <i>STM32F103C8TX</i>	44
Figura 12 – Placa de desenvolvimento do modelo <i>Raspberry Pi 3 Model B</i> . . .	46
Figura 13 – Chassi utilizado como base para o conjunto didático	50
Figura 14 – Motor de corrente contínua utilizado no projeto	50
Figura 15 – Módulo sensor ultrassônico <i>HC-SR04</i> (esq.) e módulo sensor barométrico <i>BMP180</i> (dir.)	52
Figura 16 – Módulo sensor giroscópio e acelerômetro <i>GY-521</i> (esq.) e sensor reflexivo <i>TCRT5000</i> (dir.)	53
Figura 17 – Módulo de comunicação <i>Bluetooth HC-05</i>	53
Figura 18 – Diagrama de blocos com as funções básicas previstas para o sistema	54
Figura 19 – Relação dos pinos de conexão à plataforma <i>Bluepill STM32F103</i> . .	58
Figura 20 – Bateria <i>Li-Po</i> para alimentação do sistema	59
Figura 21 – Circuito Amplificador de Diferença	60
Figura 22 – Conversor <i>Step-Down</i> para ajuste da alimentação do sistema	62
Figura 23 – Circuito Ponte-H com Controle PWM	63
Figura 24 – Circuito de Leitura para o Sensor Reflexivo <i>TCRT5000</i>	68
Figura 25 – Circuito recomendado para o controle de um <i>buzzer</i> passivo	73
Figura 26 – Circuito auxiliar para o <i>reset</i> do sistema	75
Figura 27 – Diagrama de esquemático para o circuito proposto	77
Figura 28 – <i>Layout</i> da PCI para o sistema proposto	78
Figura 29 – Vista superior do modelo 3D para a PCI do sistema proposto	79
Figura 30 – Vista lateral do modelo 3D para a PCI do sistema proposto	80
Figura 31 – Ajuste do sinal de <i>clock</i> para sincronia do sistema	82
Figura 32 – Seleção dos pinos de entrada e saída da plataforma <i>STM32F103C8TX</i>	83

Figura 33 – Exemplos de sinais de saída PWM para diferentes níveis de comparação	88
Figura 34 – Janela de configuração dos parâmetros para o Timer TIM1	89
Figura 35 – Janela de configuração dos parâmetros para o conversor ADC interno	91
Figura 36 – Janela de configuração dos parâmetros para a interface I^2C	95
Figura 37 – Janela de configuração dos parâmetros para a interface <i>UART</i>	96
Figura 38 – Diagrama de tempo para a leitura do módulo <i>HC-SR04</i>	100
Figura 39 – Janela de configuração dos parâmetros para o Timer TIM3	101
Figura 40 – Fluxograma da rotina de execução do <i>software</i> da aplicação	103
Figura 41 – Ambiente de edição para o <i>design</i> do aplicativo no <i>MIT App Inventor</i>	106
Figura 42 – Estrutura de blocos que representa a construção de uma tela virtual	108
Figura 43 – Ambiente de programação em blocos do <i>MIT App Inventor</i>	109
Figura 44 – Bloco de execução de evento de botão	110
Figura 45 – Controle para habilitar ou desabilitar funcionalidade do sistema	111
Figura 46 – Acompanhamento remoto do <i>status</i> da bateria do robô	112
Figura 47 – Criação de um gráfico com a variável de interesse	113
Figura 48 – Tela de início do aplicativo móvel	114
Figura 49 – Tela para configuração da conexão <i>Bluetooth</i>	116
Figura 50 – Tela com os controles de direção do robô	117
Figura 51 – Tela de acompanhamento dos estados dos sensores	118
Figura 52 – Tela com configurações do sistema	119
Figura 53 – Tela com informações sobre o aplicativo	120
Figura 54 – Montagem do sistema em matriz de contatos	122
Figura 55 – Carregador balanceador para baterias do tipo <i>Li-Po</i>	126
Figura 56 – Gravador para plataformas STM	127
Figura 57 – Kit básico de robótica para plataformas <i>Arduino</i>	129

LISTA DE TABELAS

Tabela 1 – Especificações técnicas de uma placa <i>Arduino Nano®</i>	42
Tabela 2 – Especificações técnicas de uma placa <i>Arduino Due®</i>	43
Tabela 3 – Valores de venda para as placas <i>Arduino</i>	43
Tabela 4 – Especificações técnicas de um microcontrolador <i>STM32F103C8TX</i>	45
Tabela 5 – Especificações técnicas de diferentes modelos de <i>RaspberryPi</i>	46
Tabela 6 – Valores de venda para as placas <i>RaspberryPi</i>	47
Tabela 7 – Protocolo de comunicação com os comandos para o sistema	98
Tabela 8 – Protocolo de comunicação com os eventos gerados pelo sistema	99
Tabela 9 – Tabela de custos do projeto	124
Tabela 10 – Estimativa de custos extras para o protótipo	125
Tabela 11 – Comparativo de custos do projeto com os <i>kits</i> de referência	128
Tabela 12 – Lista de componentes para a construção da PCI	148

SUMÁRIO

1	INTRODUÇÃO	13
1.1	Objetivos do Trabalho	14
1.2	Organização deste estudo	15
1.3	Motivação	16
1.3.1	Dificuldades no processo de aprendizado	16
1.3.2	Limitações financeiras do uso da tecnologia	20
1.3.3	O contexto da Robótica Educacional	22
2	FUNDAMENTAÇÃO TEÓRICA	27
2.1	Soluções comerciais	27
2.1.1	LEGO Mindstorms®	27
2.1.2	TI-Robotics-System-Learning-Kit®(TI-RSLK)	31
2.1.3	Makeblock®	33
2.1.4	Sphero®	37
2.2	Projetos similares	39
2.2.1	N-Bot	39
2.2.2	Praxedes	40
2.2.3	RoboFácil	40
2.3	Possíveis cenários de implementação	41
2.3.1	Arduino	41
2.3.2	Bluepill STM32F103C8TX	44
2.3.3	Raspberry Pi	45
3	PROPOSTA DE PROJETO	48
3.1	Descrição geral	48
3.2	Especificações	48
3.2.1	Estrutura mecânica	48
3.2.2	Hardware	51
3.2.3	Firmware e Software	54
3.2.4	Aplicativo para Dispositivos Móveis	55
4	DESENVOLVIMENTO DO SISTEMA PROPOSTO	56
4.1	Considerações Gerais	56
4.2	Desenvolvimento do Hardware Eletrônico para o Sistema Proposto	56
4.2.1	Definições do Diagrama Esquemático do Sistema Proposto	56
4.2.1.1	O microcontrolador	57
4.2.1.2	Especificação da Bateria e Regulador de Tensão	58

4.2.1.3	Acionamento dos Motores CC	62
4.2.1.4	Módulos de sensores	64
4.2.1.5	Circuito Seguidor de Linha	68
4.2.1.6	Circuitos Indicadores de Sinalização Visual e Sonora	70
4.2.1.7	Circuitos auxiliares	74
4.2.2	Desenvolvimento do <i>Layout</i> de PCI do Sistema Proposto	76
4.3	Desenvolvimento de <i>Firmware</i> e <i>Software</i>	80
4.3.1	Inicialização do Sistema	81
4.3.2	Configuração dos <i>GPIO</i>	83
4.3.3	Temporizações	84
4.3.4	Acionamentos de Motores CC por Modulação PWM	86
4.3.5	Leituras dos canais do ADC	90
4.3.6	Comunicação pela interface <i>I²C</i>	94
4.3.7	Comunicação pela interface <i>UART</i>	96
4.3.8	Leitura do sensor ultrassônico	99
4.3.9	Gravação do Programa e Cuidados Adicionais	103
4.4	Desenvolvimento de Aplicativo para Dispositivos Móveis	105
4.4.1	Interface Gráfica de Usuário	106
4.4.2	Ambiente de Programação em Blocos	109
4.4.3	Organização dos eventos de interface	110
4.4.4	Descrição Geral das Funcionalidades do Aplicativo Móvel	114
5	RESULTADOS OBTIDOS	121
5.1	Testes Finais	121
5.2	Análise de custos	122
5.3	Comparação com sistemas de referência	127
6	CONSIDERAÇÕES FINAIS	131
6.1	Análise dos resultados	131
6.2	Propostas de projetos futuros	133
	REFERÊNCIAS	135
	APÊNDICES	139
	APÊNDICE A – IMPLEMENTAÇÕES DE CÓDIGO PARA O SISTEMA EMBARCADO	140
	APÊNDICE B – LISTA DE MATERIAIS (<i>BILL OF MATERIALS - BOM</i>) DO CONJUNTO DIDÁTICO PROPOSTO	148

1 INTRODUÇÃO

Atravessando um período histórico de mudanças constantes, a sociedade moderna vê-se cercada por um turbilhão de informações, disparadas de todos os lados. Mesmo em grande volume, essa massa de informações ainda não é capaz de atingir todas as camadas da população, tornando o acesso (ou a falta dele) apenas mais um agente excludente. Nesse contexto, acompanhada por situações de grandes dificuldades socioeconômicas e realidades desproporcionais, a desinformação toma conta e os abismos passam a se alargar cada vez mais. Embora a resposta mais óbvia pareça ser a distribuição direta de ferramentas tecnológicas à população (uma avaliação bastante razoável), ter acesso a essas soluções não é necessariamente o último estágio na derrubada dessas barreiras políticas e culturais.

Mesmo que grande parte da população esteja bastante acostumada com a evolução tecnológica no âmbito comercial e publicitário, graças ao advento do uso de dispositivos portáteis multifuncionais, ainda é pequena a parcela desse grupo que possui contato direto e pessoal com a tecnologia, com domínio suficiente para não ser limitado por uma janela cuidadosamente criada para experiência de usuário. Uma quantidade expressiva de usuários só tem alcance das plataformas digitais até o ponto que a acessibilidade permite, não conseguindo lidar com sistemas menos intuitivos. Uma das causas para esse problema pode residir na maneira como essas formas de tecnologia são introduzidas na vida desses indivíduos, muitas vezes exigindo apenas interações básicas e fazendo o máximo para que o usuário tenha de fazer o mínimo.

Acompanhando esse cenário delicado, o ambiente educacional possui um papel fundamental na aproximação dessas ferramentas, pois, mesmo que não consiga atingir diretamente pessoas que não estejam inseridas no cotidiano acadêmico, permite, aos que ainda frequentam, a capacidade de melhorar e desenvolver habilidades motoras e cognitivas de maneira atrativa e intuitiva, e, com uma visão esperançosa, ainda transmitir esses conhecimentos a quem não os consegue acessar. Diante disso, vale lembrar da geração que já nasceu totalmente imersa em um ambiente muito próximo ao virtual, desenvolvendo naturalmente mais habilidades voltadas às plataformas de interação além do meio físico, sentindo-se muito mais à vontade circulando por rotas digitais ao invés de se limitar ao mundo físico e suas dinâmicas tradicionais. Nesses casos, o fato de a educação preocupar-se em incentivar os diferentes conceitos de inteligência associados ao mundo tecnológico, motiva o desenvolvimento de técnicas pedagógicas eficientes que funcionam como agentes motivadores para o investimento no desenvolvimento científico.

Além de todo esse contexto da alfabetização digital, a ideia de evolução tecnológica desperta na própria organização social diferentes possibilidades de expansão,

dando ao uso das novas tecnologias uma recontextualização, atuando como uma saída para muitos dos problemas vivenciados no cotidiano. Torna-se cada vez mais comum a prática de ofícios e o desenvolvimento de carreiras profissionais que lidam com a criação e aplicação dessas invenções, nas mais diversas áreas de conhecimento.

Como consequência, um dos problemas mais inflados por essa dificuldade de acesso à informação é ainda mais aquecido: até o momento de ter uma carreira profissional estabelecida, é preciso antes ter acesso a informações que tragam o conhecimento necessário. Para poder entender sobre a tecnologia e trabalhar com ela, é preciso conhecer a mesma, ter acesso ao seu funcionamento mais básico e domínio sobre suas particularidades. Portanto, é reforçado o papel de instituições de ensino voltadas ao fomento da educação tecnológica como agentes moldadores dos próprios processos de evolução desses meios. Em um cenário utópico, a solução seria simples: discutir e aplicar a tecnologia em sala de aula para desenvolver indivíduos prontos para essa nova organização social. Na prática, a situação é outra.

Não é suficiente apenas a vontade de estruturar sistemas de aprendizado capazes de apresentar conteúdos de caráter tecnológico, permitindo a introdução adequada desse tema nos meios sociais, é necessário também lutar contra as próprias barreiras socioeconômicas que limitam a difusão da informação técnica e atuam como inibidoras da construção de um ferramental suficiente para a discussão desses temas.

1.1 Objetivos do Trabalho

Como maneira de apresentar uma discussão sobre os caminhos do ensino de robótica na realidade das instituições de ensino brasileiras, este trabalho tem por objetivo construir uma alternativa prática para o desenvolvimento de ferramentas capazes de interferir positivamente nas dificuldades econômicas que cercam as barreiras educacionais. Avaliando o viés social, o objetivo traçado aqui possui embasamento em uma série de breves discussões sobre o impacto tecnológico na vida humana, a dificuldade de manutenção da vida acadêmica e as barreiras enfrentadas por aqueles que ainda tentam fazer da democratização dessas plataformas sua abordagem de trabalho.

Em um contexto geral, este trabalho tem por objetivo desenvolver uma plataforma didática de baixo custo e fácil construção artesanal, apresentando detalhadamente cada elemento que compõe o sistema e desenvolvendo uma documentação de todos os processos envolvidos na confecção do protótipo final. Ainda, com o intuito de propor uma solução aos problemas cotidianos com foco na viabilidade econômica, é uma preocupação desta pesquisa procurar o máximo de ferramentas acessíveis de baixo ou nenhum custo, que auxiliem e facilitem toda a construção do *kit* de desenvolvimento para atividades em robótica.

Para alcançar os objetivos gerais que envolvem a criação de um *kit* didático, alguns objetivos intermediários devem ser cumpridos:

- Desenvolver o diagrama de esquemático elencando os módulos e componentes discretos necessários para a implementação das funções básicas do *kit* de robótica;
- Construir um modelo de *layout* para a confecção de uma placa de circuito impresso que servirá como base para o desenvolvimento da solução;
- Selecionar adequadamente uma plataforma microcontrolada que seja capaz de automatizar e gerenciar as funções básicas do robô;
- Idealizar uma aplicação embarcada por meio linguagem compatível com a plataforma escolhida, responsável por controlar as funções mais básicas do sistema, como acionamento dos motores, atualização dos sensores e comunicação com entidades externas;
- Buscar por maneira acessível de desenvolver um sistema externo para comunicar com o robô, que permita o envio e recebimento de comandos, priorizando interfaces sem fio.
- Descrever detalhadamente as escolhas e decisões realizadas ao longo da construção dos subconjuntos, com documentação acessível e com foco na reprodutibilidade;
- Confeccionar artesanalmente e testar as implementações desenvolvidas, garantindo a correta execução de todas as funções esperadas para o *kit*;

1.2 Organização deste estudo

Este estudo desenvolve-se em cinco capítulos, com discussão teórica e embasamento acadêmico da problemática da educação tecnológica brasileira, seguido de uma avaliação geral do projeto, que culmina na definição de uma possível solução para a democratização digital.

Ao longo do capítulo 1, é apresentada uma sequência de análises socioeconômicas sobre a relação das pessoas com os cursos de ensino superior, a inserção da tecnologia nesses meios de produção intelectual, assim como o contato cotidiano com as temáticas de fundamentos da robótica. O objetivo principal deste capítulo é fornecer uma ideia clara sobre os percalços atualmente enfrentados e como o incentivo ao conhecimento desses avanços pode ser um fator bastante positivo no enfrentamento das barreiras culturais.

Já o capítulo 2, apresenta maneiras atualmente disponíveis para suprir essas lacunas da robótica didática, seja por meio de conjuntos comerciais proprietários ou iniciativas de cunho acadêmico, como forma de retribuição social. Neste capítulo, são abordadas várias discussões sobre variações de custo e viabilidades de implementação das diferentes alternativas propostas.

Entre os capítulos 3 e 5 desenrola-se a confecção propriamente dita da solução funcional e de baixo custo, que é o produto deste estudo. Toda a parte documental e de embasamento teórico é descrita aqui de maneira simples e facilitadora para o entendimento dos processos criativos.

Por fim, o texto acompanha a conclusão do projeto e a discussão sobre os objetivos alcançados, avaliando os requisitos estabelecidos e sugestões para desenvolvimentos futuros.

1.3 Motivação

Este projeto tem o objetivo de discutir algumas das limitações vinculadas à democratização da tecnologia e evidenciar soluções a problemas enfrentados dentro e fora das salas de aula e instituições de ensino. Trata-se de um tema delicado e pertinente à atual estrutura da educação e da própria organização social em que se insere, funcionando como ferramenta de enfrentamento a dificuldades cotidianas diversas. Nos tópicos a seguir, alguns argumentos serão apresentados, com informações de viés analítico e estatístico, de maneira a guiar o desenvolvimento deste estudo.

1.3.1 Dificuldades no processo de aprendizado

Os problemas enfrentados pelo setor educacional brasileiro ultrapassam os limites pedagógicos e esbarram de maneira expressiva em contextos políticos e econômicos, influenciando no distanciamento das pessoas dos ambientes educacionais, apresentado principalmente na impossibilidade, para uma grande parcela da população, de investir em uma carreira acadêmica, seja pela dificuldade no ingresso ou insustentabilidade na permanência. Reflexos de falhas nos aspectos técnicos na construção do cenário educacional atual, são cada vez mais alarmantes os dados estatísticos que resumem a quantidade de estudantes com acesso direto ao ensino superior, mostrando um expressivo número de pessoas deixando de concluir seus cursos, abandonando-os ainda em fases iniciais. Tomando alguns dados estatísticos, é notável que a situação exige bastante atenção, ao visualizar que

Segundo dados da OCDE (2015), em 2012, 29% dos graduados da Coreia do Sul, 28% da Alemanha e 37% da França eram oriundos das ciências naturais e engenharia, ao passo que no Brasil esse percentual era apenas de 11%. [...] Segundo dados do Censo da Educação

Superior, entre 2009 e 2014, a graduação em Direito acumulou uma evasão de aproximadamente 45% ao longo desses anos, enquanto os cursos de Matemática e os de Engenharias Metalúrgica e Mecânica apresentaram, nesse mesmo período, taxas próximas aos 90% e 60%, respectivamente (SACCARO; FRANÇA; JACINTO, 2019, p. 339).

Dentre os percalços encontrados, são vários os fatores que influenciam nessa grande evasão de graduandos e baixa taxa de conclusão dos cursos de nível superior, principalmente dos cursos de engenharia. Contando com diversas formas de acesso e ingresso, o cenário visto em instituições públicas e privadas de ensino superior expande um panorama que comprova uma extensão da própria divisão social que se estabelece fora dos limites das universidades. Os recursos pessoais necessários para o investimento em um curso superior estão muito além de simplesmente tomar decisões de longo prazo sobre a própria carreira ou a universidade que se quer ingressar: ao passo que instituições privadas exigem investimentos financeiros diretos e explícitos, ao ingressar em uma universidade pública, um alto dispêndio ainda é requerido.

Analisando, principalmente cursos que são oferecidos em modalidade integral ou mesmo nos períodos diurnos, em que a dedicação exclusiva gera um custo colateral capaz de limitar a permanência de estudantes, a escolha por vínculos empregatícios, necessários para sobrevivência, sobrepõe-se e elimina logo no início potenciais concluintes. Outro fator bastante discutido e impactante relaciona-se ao acesso da informação em sua formação básica, por muitas vezes incompleta e insuficiente, o que causa um desnível entre os conteúdos básicos e os adquiridos na jornada do ensino superior, abrindo uma lacuna no processo de aprendizagem do aluno, servindo também como barreira ou incentivo à desistência.

Como citam Saccaro, França e Jacinto (2019), em paráfrase a outros autores, a união entre os fatores da qualidade da educação de base, somada à própria visão do estudante sobre sua capacidade de aprendizado, pode refletir em menores desempenhos nos semestres iniciais e desencadear o processo de desistência. Em concordância, a falta de agentes ativos de moderação e motivação, agindo como contraponto à sensação de baixo rendimento, acaba por reforçar e atuar como estímulo para o crescimento acelerado das estatísticas.

Ainda, Saccaro, França e Jacinto (2019) apontam resultados de estudos realizados com uma amostra de estudantes de engenharia, discutindo que alunos que dedicam mais de 10 horas semanais em atividades desvinculadas do curso, como empregos fixos, apresentavam maior probabilidade de reprovação em disciplinas. Outro argumento que se alinha à falta de motivação diz respeito a estudantes que classificaram seu conhecimento básico abaixo da média ou mesmo filhos de pais que não possuem diploma de ensino superior, os quais demonstravam maior propensão de abandono do curso durante o processo. Surge então a discussão sobre a eficiência

e necessidade de ações afirmativas capazes de estreitar essa relação de estudantes socialmente desestimulados a seguirem suas carreiras acadêmicas.

Mencionando principalmente as ações voltadas ao âmbito acadêmico, como sistemas de bolsa, financiamentos estudantis e reservas de vagas de ingresso, nota-se a presença desses como facilitadores da democratização do ensino, mesmo que não sejam capazes de mitigar todas as vulnerabilidades, auxiliam no processo motivacional dessa jornada acadêmica. Por meio dos estudos de Saccaro, França e Jacinto (2019), citando apontamentos de MENDES JUNIOR (2014), explicitou-se que este:

(...) avalia a evasão de alunos cotistas da Universidade do Estado do Rio de Janeiro – primeira instituição de ensino superior do país a implementar o sistema de cotas. Entre os seus resultados, o autor aponta que esses estudantes apresentaram um Coeficiente de Rendimento pior ao longo do curso, quando comparados com alunos não cotistas. Porém, alunos contemplados com essa política de ação afirmativa apresentaram maiores taxas de graduação que os demais, indicativo de que esses alunos dão elevado valor aos cursos nos quais ingressam (SACCARO; FRANÇA; JACINTO, 2019, p. 345).

Como argumento que reforça a grande pressão social que sofrem as classes sociais mais deficitárias, verificar que alunos agraciados com ações afirmativas dão mais valor aos cursos, considerando a educação como ferramenta de expansão e conquista de espaço na sociedade, quebrando as barreiras historicamente estabelecidas, permite compreender um panorama maior sobre os critérios de desistências e pontos fracos dos sistemas de ensino. A aplicação de políticas públicas de incentivo à educação além dos limites da sala de aula é, sem dúvidas, uma alternativa essencial na democratização do conhecimento e no combate à deficiência educacional que enfrentamos. Em resumo,

A ação afirmativa se diferencia das políticas puramente antidiscriminatórias por atuar preventivamente em favor de indivíduos que potencialmente são discriminados [...] No debate público e acadêmico, a ação afirmativa com frequência assume um significado mais restrito, sendo entendida como uma política cujo objetivo é assegurar o acesso a posições sociais importantes a membros de grupos que, na ausência dessa medida, permaneceriam excluídos. Nesse sentido, seu principal objetivo seria combater desigualdades e dessegregar as elites, tornando sua composição mais representativa do perfil demográfico da sociedade (GEMAA, s.d).

De outro lado, acreditar que apenas as ações afirmativas são suficientes para erradicar completamente a discriminação sofrida por minorias vulneráveis é completamente ilusório. Nesse ponto, a educação como ferramenta deve estabelecer abordagens eficientes que funcionem como políticas anti-discriminatórias, realizando as obrigações básicas de um sistema que deve atingir igualmente todas as esferas sociais.

Entretanto, não apenas as barreiras sociais são um problema na realidade do ensino superior: além de dificuldades socioeconômicas, ainda existem situações que discutem a construção da cadência de conteúdos apresentados ao longo da jornada de aprendizado, que acabam por afastar o interesse do aluno. Falta de acesso a materiais diferenciados e limitações na contextualização prática das atividades são grandes fatores que geram no graduando a não-identificação ou mesmo decepção na relação expectativa/realidade alimentada previamente.

Muito além de apenas aceitar alunos ingressantes nas mais diversas áreas de atuação da educação em vertentes tecnológicas, uma das maiores preocupações da estrutura interna dos cursos é garantir que estes estudantes consigam alcançar seu diploma ao final do processo. Entretanto, na prática, o cenário encontrado nesse aspecto é bastante complicado, com uma parcela expressiva de alunos deixando suas vagas sem obter seu título de graduação.

Abordando mais especificamente a área das ciências exatas, na qual se encaixam cursos baseados no desenvolvimento tecnológico, estudos comprovam que o período com maior evasão de graduandos são os primeiros semestres, por apresentar disciplinas de caráter mais teórico, aumentando o coeficiente de desistência ao inflar o sentimento de que o curso não se encaixa nas expectativas. Por serem de grande importância no contexto geral, para construir a carga necessária ao entendimento de conteúdos futuros, é um grande desafio criar um plano de ensino que consiga tornar mais atrativa essa etapa de aprendizado. (FREGONESIS, 2002)

Além disso, o ser humano, como ser social, passa a se identificar mais com um grupo ao se sentir parte dele. Algo parecido ocorre com o ambiente acadêmico, no qual estudantes mais engajados desfrutam da sensação de pertencimento do contexto em que tenta inserir-se, quase uma prévia da próxima etapa almejada na sucessão da vida acadêmica: a inserção no mercado de trabalho. Entretanto, a disponibilidade de tempo para realização dessas atividades extracurriculares esbarra nas adversidades socioeconômicas citadas anteriormente, aumentando ainda mais a vulnerabilidade de algumas camadas da população: se não existe tempo disponível, não existe integração com o curso. É um estigma difícil de ser combatido.

Citando novamente os setores da educação que dependem de ferramentas tecnológicas, um dos principais limitantes para o desenvolvimento de atividades de aprendizado de maneira prática encontra-se na dificuldade de adquirir tecnologias de custo compatível e de acesso prático, de distribuidores nacionais e por um preço que caiba na realidade das instituições brasileiras. Nesse aspecto, o tópico a seguir discute brevemente a realidade do consumo de tecnologia no cenário nacional.

1.3.2 Limitações financeiras do uso da tecnologia

A ideia de incorporar a tecnologia no dia a dia do contexto educacional é de longe muito honrosa e extremamente promissora. Nesse caminho, umas das grandes barreiras a ser superada tem caráter financeiro e é quase intransponível. Não apenas o alto custo dos equipamentos é questionado, como também a disponibilidade em âmbito interno, no mercado nacional.

Já é bastante comum a professores, estudantes e entusiastas de tecnologia ter certa dificuldade em encontrar os componentes adequados para a criação de seus próprios projetos. Com um número reduzido de distribuidores internos, a busca por insumos, mesmo por meio de *e-commerce*, ainda é um processo dificultoso. Não apenas o pequeno número de fornecedores assusta quem depende destas tecnologias, mas a pouca variedade de produtos acaba limitando o desenvolvimento de soluções, tirando a possibilidade de teste em diferentes cenários de implementação.

Para suprir essa lacuna, outra opção, bastante atrativa, dá-se pela globalização, com a alternativa de importação dos produtos que se encontram indisponíveis em território brasileiro ou tecnologias que ainda não adentraram o cotidiano nacional. As opções são diversas, já que a maioria dos distribuidores dos poucos produtos que são acessíveis no mercado interno são provenientes de empresas estrangeiras, de países como China e Estados Unidos. A falta de estímulo para esse tipo de investimento vem do alto custo: a maioria dos produtos que entram no país passam por processos burocráticos de avaliação e podem ser duramente taxados pela Receita Federal.

Ao consultar a legislação vigente, é fácil identificar uma série de tributações aplicadas sobre produtos importados quando estes atravessam as barreiras aduaneiras brasileiras. Algumas ferramentas disponibilizadas pelos ambientes virtuais da Receita Federal permitem estimar estes impostos, como é o caso do Simulador do Tratamento Tributário e Administrativo das Importações, que calcula diretamente cada uma das taxas cobradas para estes itens, acessando automaticamente as alíquotas relacionadas a cada um. As proporções de tarifas aplicadas a um produto são categorizadas de acordo com a Tabela de Incidência do Imposto sobre Produtos Industrializados, a TIPI, que descreve todos as categorias e subcategorias de produtos industrializados, de acordo com a Nomenclatura Comum do Mercosul (NCM), atribuindo as taxas ao produto de acordo com a sua categoria.

Reunindo todos esses critérios, o simulador recebe como entrada o código NCM do produto importado, o chamado Valor Aduaneiro (que é efetivamente o custo da mercadoria), e a moeda utilizada na transação de aquisição do item. Consultando alguns sites de comercialização de produtos eletrônicos, foi verificado que *kits* de desenvolvimento em robótica podem ser atribuídos ao código 9503.00.60, que identifica "Outros conjuntos e brinquedos, para construção" dentro da categoria "Brinquedos, jo-

gos, artigos para divertimento ou para esporte; suas partes e acessórios" (MINISTÉRIO DA ECONOMIA Receita Federal, 2017).

Tomando como base uma mercadoria vinculada a este código e com valor de custo de US\$ 100,00, a uma taxa de câmbio para o dólar de R\$ 5,5830, o custo convertido diretamente deste produto é de R\$ 558,30, acrescido de R\$ 97,70 pelo Imposto de Importação - II com alíquota de 35%, com mais R\$ 37,69 para o Imposto sobre Produto Industrializado (IPI) sob alíquota de 10%, além de uma taxa de R\$ 5,86 de contribuição ao fundo PIS (Programas de Integração Social) sob alíquota de 2,10% e uma cobrança de R\$ 26,94 para o fundo COFINS (Contribuição para Financiamento da Seguridade Social) sob alíquota de 9,65%. Somando todas as tributações incidentes, um produto com custo de US\$ 100,00 no exterior chega ao Brasil com valor próximo a R\$ 726,49, segundo a simulação oficial da Receita Federal (MINISTÉRIO DA ECONOMIA Receita Federal, s.d.b).

Vale lembrar que essas tributações levam em conta a importação de produtos por meios legais e ainda desconsideram cobranças relacionadas a transporte, despesas com empresas facilitadoras de importação e ainda taxas nacionais imputadas a pessoas jurídicas pela comercialização destes produtos. Esse cenário corrobora tanto o alto custo de mercadorias que precisam ser importadas, quanto a baixa diversidade na oferta destes bens de consumo, inviabilizando a venda deles em território nacional.

Outra opção de importação é o Regime de Tributação Simplificada (RTS), que prevê, para encomendas internacionais de até US\$ 3.000,00, o pagamento de um imposto único sob alíquota de 60%, independentemente da classificação tarifária dos bens importados. Outro ponto do regime dita que compras de até US\$ 50,00 terão abatimento dos impostos aplicados, desde que o remetente e o destinatário sejam pessoas físicas (MINISTÉRIO DA ECONOMIA Receita Federal, s.d.a). De todo modo, a importação segue dificultada, pois artefatos de tecnologia costumam ter valores superiores a US\$ 50,00 e uma taxa fixa de 60% sobre essas mercadorias eleva drasticamente os custos de aquisição.

Essas limitações deixam ainda mais distantes as possibilidades de investimentos em equipamentos tecnológicos, que, por sua complexidade construtiva, possuem um elevado custo e conseqüentemente valor comercial, influenciando o cenário educacional, que passa a não contar com laboratórios densamente equipados e materiais adequados às necessidades dos alunos.

Falta de investimentos e altos cortes nos fundos destinados à educação pressionam os setores de coordenação a realizar difíceis escolhas nos investimentos cotidianos, abrindo mão da possibilidade de ter equipamentos de ponta em troca de condições mínimas de manter aberta a oferta de vagas e o funcionamento das instituições com pouquíssimos recursos. Em um suspiro de esperança, resta aos educadores

encontrar formas criativas de fazer muito com o pouco que se tem, investindo em projetos de pesquisa que permitam a criação de equipamentos substitutos para aqueles que não podem ser adquiridos.

1.3.3 O contexto da Robótica Educacional

Segundo Zilli (2004), o conceito da Robótica Educacional, também referida como Robótica Educativa, apresenta-se como uma ferramenta disponível ao educador como artifício de auxílio na demonstração prática de conteúdos que, na teoria, são de difícil compreensão. Ainda, acrescenta que esta é "uma proposta educacional, apoiada na experimentação e na errância", incentivando o conhecimento empírico em sobressalência à clássica transmissão teórica de conhecimento.

Faz parte da natureza humana adquirir o conhecimento por meio da observação e da experimentação e, junto ao ato de trazer situações aparentemente obscuras para a luz didática da abordagem prática, tem impactos positivos para o aluno ao permitir maior facilidade de aprendizado. Além disso, o decorrer do processo é capaz de proporcionar uma maior identificação do aluno com o resultado obtido, agindo como uma motivação, desencadeada pela conquista de uma solução utilizando seus próprios meios.

Reconhecer os métodos discutidos pela ciência da robótica é por si só positivo, pois permite analisar um cenário de melhor proveito das características intrínsecas à robótica. Além disso, Zilli (2004) ainda aponta algumas das vantagens socioeducativas de se abordar a robótica educacional, que se encontram nas atividades:

- Promover a interdisciplinaridade e o trabalho conjunto;
- Traduzir para aspectos práticos e manuais as peculiaridades de modelos de simulação;
- Tornar acessíveis os conceitos de Ciência e Tecnologia aos alunos;
- Despertar a consciência da presença da ciência na vida cotidiana;
- Incentivar a autossuficiência na resolução de problemas e obtenção de conhecimentos;
- Desenvolver habilidades de investigação e resolução de problemas concretos;

Outros aspectos bastante discutidos nos estudos relacionados à robótica educacional levam em conta as metodologias de construção do processo de desenvolvimento cognitivo, proporcionado pelos conjuntos de robótica. Dentre os estudiosos largamente citados, um se destaca em diversas situações: Seymour Papert, psicólogo

pertencente ao Laboratório de Inteligência Artificial do (MIT), o berço dos conjuntos de robótica.

Defendendo uma teoria denominada Construcionismo, Papert acreditava que a relação entre computador e aluno devia ser estabelecida de forma a dar ao aluno o poder da decisão e que este deve se posicionar como unidade ativa do conhecimento. Não é tarefa do computador conduzir o processo de aprendizado do aluno e sim funcionar com o único e exclusivo propósito de atuar como ferramenta facilitadora para o acesso ao conhecimento almejado. Assim,

Nessa perspectiva da aprendizagem em ambientes computacionais, Papert considera as “iniciativas, expectativas, necessidades, ritmos de aprendizagem e interesses individuais dos alunos”, que é incitado a estabelecer conexões entre o novo conhecimento em construção e outros conceitos de seu domínio, empregando para tal a sua intuição. (BASTOS, 2002)

Para Zilli (2004), desenvolver a visão de Papert é ter como meta a busca pelo melhor rendimento, quando o aluno consegue assimilar o máximo de conteúdo possível, com o mínimo de instrução externa. O conceito de Construcionismo baseia-se na ideia de que o incentivo ao aprendizado se dá de maneira mais eficiente quando o aluno atinge o conhecimento por seus próprios meios, sem ser diretamente direcionado, seja por auxílio de um profissional orientador, seja pela ajuda de uma máquina. A palavra de ordem do Construcionismo é a intuição.

Em seu aspecto educacional, os apontamentos realizados por Papert foram adaptados dos estudos de outro grande influenciador dos preceitos da Robótica Educacional: Jean Piaget. Esse psicólogo, biólogo e filósofo suíço propôs uma teoria de aprendizagem denominada Construtivismo, que invoca o papel ativo do aluno na busca do conhecimento citado por Papert, reforçando que as relações de interação meio-objeto devem ser entendidas por meio de vivências e experimentação. Em um processo denominado *assimilação*, o indivíduo adquire e incorpora novas informações, que passam a agregar em seu conhecimento. Ainda, o processo chamado *acomodação* diz respeito a informações anteriormente adquiridas e passam por um processo de ressignificação ou atribuição de uma nova função no espectro do conhecimento. Em um resultado final, a união dos dois processos gera o processo de *equilibração*, ou o acesso ao equilíbrio entre os conceitos antes assimilados e os recentemente adquiridos (ZILLI, 2004).

Como um terceiro olhar ao conjunto proposto, Edith K. Ackermann, também do MIT, apresentou um estudo sobre as diferenças entre as visões de Papert e Piaget, refinando o entendimento das teorias discutidas. Enquanto Piaget traduz um desenvolvimento natural das linhas de pensamento cognitivo das crianças, passando por uma transformação do raciocínio concreto para o abstrato, Papert está mais interessado na

concretização do conhecimento e materialização das ideias concebidas. Piaget trata do conhecimento sem propor a utilização dos meios, enquanto Papert atribui ao processo de execução importância máxima no aprendizado.

Em concordância, César (2020) aponta que o estudo da robótica consiste na reimaginação do ambiente e das tecnologias já disponíveis, se baseando no ímpeto construtivo, natural ao ser humano, como artifício de desenvolvimento de sua própria sensibilidade e razão. Ao atuar como proposta didática, a robótica educacional pode ser utilizada como estímulo ao ensino transdisciplinar, permitindo ao educando e aos educadores projetar a essência de seus próprios corpos nos movimentos de suas criações, tornando-as uma extensão de si mesmo.

Além disso, César (2020) reafirma que essas propostas pedagógicas "rompem com a perspectiva fragmentada e compartimentalizada do currículo escolar, pois trazem para a discussão temas que transversalizam diferentes áreas do conhecimento". O conceito por trás da robótica educacional não deve ser entendido como apenas relacionado aos artefatos físicos e as tecnologias utilizadas como ferramenta, mas referir-se ao conjunto completo das propostas de intervenção e procedimentos que fazem uso de dispositivos robóticos como mediadores da construção do conhecimento.

Pensando no cenário atual, a robótica utilizada no cenário educacional é aplicada como forma de "treinamento" dos educandos, ao fazer uso de plataformas padronizadas, adquiridas comercialmente e constituídas principalmente por *softwares* e *hardwares* não-livres. De maneira oposta, é possível pensar no conceito de Robótica Pedagógica Livre (RPL), ou simplesmente Robótica Livre (RL), que define os objetivos de ensino baseados na necessidade e planejam o processo educacional como oportunidade de integração, mudança de comportamento e conduta, por meio de plataformas que seguem quatro pilares de liberdade: a liberdade de uso, de estudo e modificação, de distribuição e de redistribuição das melhorias (CÉSAR, 2013).

Essas ideias de liberdade já são comuns ao mundo dos *softwares*, entretanto, para os *hardwares*, as condições tecnológicas e financeiras impedem a customização completa de todas etapas, dificultando, por exemplo, o projeto de um componente eletrônico discreto. Assim, para driblar essas impossibilidades, a robótica livre propõe o uso de *softwares* livres como base de programação e a integração de sucata de equipamentos eletrônicos e/ou insumos acessíveis para a construção física destes artefatos cognitivos (CÉSAR, 2013). Não apenas trazer acessibilidade, a ideia da robótica livre é apresentar ao educando todas as etapas atreladas ao desenvolvimento das soluções, navegando por diferentes níveis de abstração do conhecimento.

Ao passo que a robótica educacional como conceito passou a desenvolver-se, outras vertentes de organização de ensino passaram a figurar nas discussões, passando a dividir espaço e até mesmo confundir-se com os conceitos aplicados a essa

ciência. Um exemplo muito forte de modelo de visualização, surgido no início da década de 1990, porém, popularizado apenas por volta do ano de 2001, o modelo de ensino *STEM*, do inglês *Science, Technology, Engineering and Mathematics* ou, em tradução, "Ciência, Tecnologia, Engenharia e Matemática", apresentou-se como uma corrente de pensamento disruptiva em busca de uma nova abordagem que pudesse opor-se às limitações de métodos tradicionais de ensino. Essa quebra da tradicionalidade busca reposicionar os estudantes dessas áreas da ciência, tirando-os da posição passiva, sem interação com o objeto de estudo, e colocando-os em ambientes que estimulem a experimentação e o aprendizado empírico (PUGLIESE, 2017).

Nesse processo, Pugliese (2017) cita que alguns autores propõem a necessidade de adição de outro pilar de conhecimento à sigla, baseado na educação artística (*Arts*), passando a ser denominada *STEAM*. A inclusão dessa vertentes é justificada como um equilíbrio e a inserção de um contexto sociocultural ao aprendizado, deixando o modelo mais aberto ao ambiente em que o estudante está inserido, vinculando essas subjetividades em seu desenvolvimento científico.

É importante mencionar que, assim como diversos aspectos culturais globais, a proposição do método *STEM* de ensino teve início no Estados Unidos e, além de ter a intenção de dialogar com as transformações tecnológicas pelas quais a sociedade passou, a criação deste novo modelo pretendia deixar o estudo de ciências mais atrativo, como resposta à falta de profissionais nessas áreas de estudo. "Esse fator é um dos mais relevantes para compreender o movimento. Está em sua gênese o senso de crise nas áreas *STEM* e a ideia de que essas áreas são vitais para a era tecnológica na qual as sociedades estão inseridas atualmente"(PUGLIESE, 2017). É evidente que o foco do modelo tem visão no retorno econômico que o aquecimento dessas áreas de interesse refletem, além de sustentar sua posição competitiva no mercado, mostrando preparação para lidar com uma sociedade moderna e tecnologicamente avançada.

Pulando para o cenário nacional, as iniciativas que relacionam esse modelo como base para o desenvolvimento de atividades de fomento à ciência ainda não possuem grande impacto e influência, com uma disseminação bastante reduzida. Entretanto, isso não significa que o movimento passou despercebido ou mesmo que não influenciará em decisões futuras, já que algumas iniciativas privadas e até mesmo ações governamentais passaram a incentivar a busca por esses valores, criando programas que incentivaram a importação desses mecanismos para o cenário educacional interno (PUGLIESE, 2017).

Com essa visão, o papel deste movimento seria de trazer uma renovação no cenário educacional brasileiro, evidenciando aspectos tecnológicos em concordância com o crescimento do potencial produtivo do país. Além disso, por não investir em força de exportação de propriedade intelectual, seguindo a corrente global, é natural

o processo de absorção destes modelos de fontes externas que já validaram sua eficiência.

Trazendo estes conceitos para o contexto do mundo real, o desenvolvimento de processos de aprendizado que se preocupam mais com as evoluções gradativas do indivíduo antes da execução de um produto final, pode refletir na promoção de ambientes de incentivo ao crescimento destas áreas de interesse, atuando como agentes de democratização da informação. Uma das aplicações práticas da implantação destes cenários lúdicos, por meio da robótica, são as diversas competições dedicadas ao gênero, de alcance nacional e internacional, que reúnem aspirantes e estudiosos dedicados, focados em um objetivo comum.

Um exemplo bastante interessante para esse tipo de iniciativa é a Competição Brasileira de Robótica (CBR), considerada a maior competição de robótica voltada a estudantes universitários do Brasil, tendo reunido mais de 200 instituições e 1500 inscritos na edição de 2019. Com incentivo e apoio do *Institute of Electrical and Electronic Engineers (IEEE)*, o evento é destinado a estudantes de ensino superior, entre graduação e pós-graduação e pesquisadores de área, propondo a seus participantes desafios que tentam representar problemas cotidianos e devem ser resolvidos por robôs de comportamento autônomo (ROBOCUP BRAZIL, c. 2019).

Paralelamente ao evento da CBR, ocorre a final de outra competição bastante relevante para os aficionados por robótica: a Olimpíada Brasileira de Robótica (OBR). Com seu caráter público e gratuito, a OBR é uma competição para estudantes de escolas públicas e privadas que cursam ensino fundamental, médio ou técnico, com o intuito de levantar questionamentos e discussões voltadas ao ensino do tema em escolas de educação básica do país todo (OBR, c. 2020).

2 FUNDAMENTAÇÃO TEÓRICA

Com o crescimento acelerado da indústria voltada à aplicação dos princípios e funcionalidades da robótica no cotidiano, não demorou a surgir um nicho que se aproveitasse dos aspectos recreativos das soluções para investir pesado em linhas de produto voltadas a *kits* de desenvolvimento. Prometendo incentivar seus consumidores a adentrar esse novo campo da criatividade, o mercado viu surgir uma série de projetos com aspectos que se aproveitam da Robótica Educacional e tentam aproximar plataformas bastante populares aos ambientes de aprendizado.

Em contrapartida, sofrendo com os problemas socioeconômicos visitados anteriormente, diversos núcleos internos às próprias instituições de ensino difundem seu conhecimento na busca por alternativas mais acessíveis. Sendo familiares aos cenários economicamente desproporcionais apresentados pelo mercado comercializador de *kits* didáticos, diversos são os projetos que investigam abordagens de aplicação e mecanismos de melhor aproveitamento dessas plataformas, focando nas mais diversas áreas de interesse e almejando o alcance interdisciplinar idealizado pela Robótica Educacional.

2.1 Soluções comerciais

Como efeito desse grande avanço tecnológico, várias empresas, tanto de pequeno como de grande porte, investiram nesses produtos, buscando atrair um público sedento por possibilidades de agregar e integrar a eletrônica em suas atividades e diversões diárias. Para contextualizar empresas e produtos que obtiveram êxito nessa tarefa, os tópicos a seguir apresentam diferentes exemplos que se utilizam da robótica como objeto de ensino.

2.1.1 LEGO Mindstorms®

Era 1984, o então *CEO* do grupo *LEGO*, Kjeld Kirk Kristiansen, assistia um programa de televisão quando se deparou com uma entrevista do professor do *MIT*, Seymour Papert, demonstrando o uso da linguagem *LOGO* como artifício para controlar robôs em padrões pré-definidos. Dentro da própria *LEGO* existia há alguns anos uma divisão voltada ao estudo e aplicação de atividades educacionais como a linha *Technic*, que já apresentava princípios de mecânica e pneumática em seus brinquedos. A curiosidade pelas possibilidades de expansão das funcionalidades aos produtos *LEGO*, quando incorporadas com as teorias e técnicas de aprendizagem desenvolvidas por Papert, fez com que não demorasse o arranjo de uma visita de Kristiansen aos laboratórios do *MIT*, imediatamente estabelecendo uma forte parceria (SILVA, 2018).

Segundo Watters (2015), esse processo não foi uma surpresa aos integrantes do *MediaLab*, laboratório conduzido por Papert, já que haviam estudos em andamento sobre a integração da linguagem *LOGO* a sistemas de peças *LEGO*, mesmo antes do surgimento de interesse por parte de Kristiansen. O projeto *LEGO-LOGO* queria ir além dos comandos pré-definidos apresentados por Papert em sua entrevista e permitir que a criatividade realmente fluísse e os limites só dependessem daqueles que manipulassem o conjunto idealizado. Inicialmente, os projetos apresentaram versões apenas em ambientes escolares e de teste, com funcionalidades reduzidas. No ano de 1987, um protótipo funcional já estava pronto, entretanto, a comercialização tardou mais alguns anos.

Figura 1 – Bloco configurável da primeira geração do *LEGO Mindstorms®*



Fonte: <www.scitechexplorer.weebly.com>.

Foi então, no ano de 1998, que a maior companhia de comercialização de brinquedos de encaixar decidiu que já era momento de adentrar nesse novo nicho de mercado com força total, apresentando um produto completamente inovador, tanto em seu apelo comercial quanto no desenvolvimento científico que proporcionava, completamente alinhado aos novos horizontes da expansão tecnológica que vinha se estabelecendo no mundo da robótica recreativa. E foi assim que surgiu o que hoje é bastante conhecido como *LEGO Mindstorms*, kit de desenvolvimento de robótica que une as já famosas peças de encaixar *LEGO* com sistemas microprocessados e blocos

adaptados a motores, sensores e estruturas de automação. Já em seu lançamento, o conjunto foi um sucesso de vendas e pelo valor US\$ 200,00 era possível adquirir um total de 717 peças de diferentes formatos e funções, além do acesso ao software com interface de interação com o produto. O núcleo de processamento era concentrado em um bloco programável chamado "RCX", com design inspirado nas próprias peças *LEGO*, demonstrado na Figura 1.

Completamente abraçado por seus consumidores e considerado pelo *The New York Times* uma revolução para uma companhia, que vinha sofrendo com a perda de espaço em frente à entrada de cada vez mais brinquedos eletrônicos no mercado, não demorou muito para que até mesmo grupos de engenheiros profissionais dedicassem esforços para popularizar, na recentemente estabelecida rede mundial de computadores, *sites* com sugestões e guias, tutoriais e códigos úteis para a criação e programação do conjunto em diferentes formatos, expandindo ainda mais o alcance da plataforma e surpreendendo até mesmo a própria companhia *LEGO* com a repercussão. Não demorou também para que outras companhias apresentassem soluções compatíveis ao *LEGO Mindstorms*, como sistemas operacionais adequados ao processador empregado (VALLANCE et al., 2009).

Com o decorrer dos anos, o sucesso em seu meio de comercialização fez com que o *LEGO Mindstorms®* tivesse outras gerações de produto, com especificações técnicas melhoradas ao longo do tempo: depois da primeira geração, em 2006 foi lançado o modelo *Mindstorms NXT®* e, em 2013, o atual modelo comercializado, *Mindstorms EV3®*, com os blocos configuráveis de ambos apresentados na Figura 2.

Figura 2 – Blocos configuráveis da segunda (esq.) e terceira (dir.) gerações do *LEGO Mindstorms®*



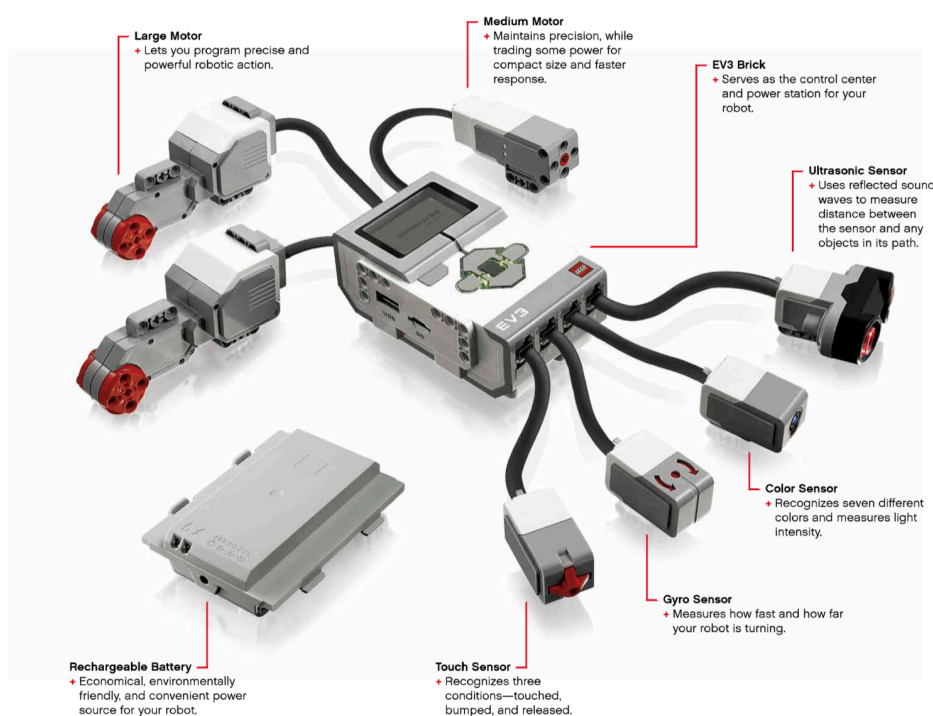
Fonte: <www.generationrobots.com>.

Citando suas especificações técnicas, o sistema automatizado do *LEGO Mindstorms®* de primeira geração contava com um processador *Hitachi H8/300* de 16 MHz e memória *RAM* de 32 kB e *ROM* de 16 kB, além de poder ser associado a dois motores, dois sensores de toque, um de luminosidade e um pequeno *display*

LCD de segmentos. Na geração seguinte, a *NXT*, o processador foi atualizado para um *ARM7* de 48 MHz, assim como para uma memória *RAM* de 64 kB, contando agora com uma memória de armazenamento em *Flash* de 256 kB e o kit passando a conter três servo motores, um sensor de toque e um sensor multifunção (luminosidade, som e distância). O *display LCD* também passou por atualização, passando para um, ainda monocromático, de 100 x 64 *pixels*. Outra novidade da segunda geração foi a introdução do suporte a *Bluetooth®* (WATTERS, 2015).

Chegando ao último modelo lançado, o *EV3*, atualmente comercializado, além da melhoria com a introdução do processador *ARM9* de 300 MHz, aliado a uma memória *RAM* de 64 MB, uma *Flash* de 16 MB, passando a suportar o uso de cartões *microSD* (WATTERS, 2015). Entretanto, além das melhorias de especificação, o bloco, que contém um sistema operacional *Linux* de natureza *open-source*, teve a adição de entradas *USB 2.0*, suporte a conexão *WiFi* e a dispositivo móveis de sistemas *Android®* e *iOS®*. Em seu conteúdo, o *kit* ainda apresenta dois motores grandes, um motor médio, dois sensores de toque, um sensor de cor, giroscópio e um sensor ultrassônico, como evidenciados na Figura 3.

Figura 3 – Bloco configurável do *LEGO Mindstorms®EV3* e seus módulos adicionais



Fonte: Adaptado de (LEGO, 2013).

Remetendo à discussão anterior, sobre o alto custo da inserção de tecnologias de ponta no mercado interno, o último modelo lançado do *LEGO Mindstorms®*,

mesmo datando de 2013, ainda é um artigo bastante inacessível ao bolso do brasileiro. Em consulta realizada no mês de abril de 2020, a loja americana oficial dos produtos *LEGO®* apresentava seu modelo *EV3* pelo valor de aproximadamente US\$ 349,99(<www.lego.com>). De mesma maneira, ao consultar a loja brasileira, o mesmo produto apresenta valor de revenda bastante elevado e fora da realidade da maioria das pessoas, em análise ao poder de compra nacional: com especificações técnicas idênticas, o modelo tinha seu preço atribuído em R\$ 2.999,99 (<www.legobrasil.com.br>).

Uma alternativa possível seria a compra do mesmo produto por meio de facilitadores, empresas que trabalham com importação de produtos ou mesmo revendedores vinculados à própria *LEGO®*. Embora exista uma pequena redução no valor final, essa solução ainda não é viável, pois depende também de flutuações de moeda e taxas adicionais de importação, tornando esse, embora um produto de referência na área, apenas mais um cenário inalcançável.

2.1.2 *TI-Robotics-System-Learning-Kit®(TI-RSLK)*

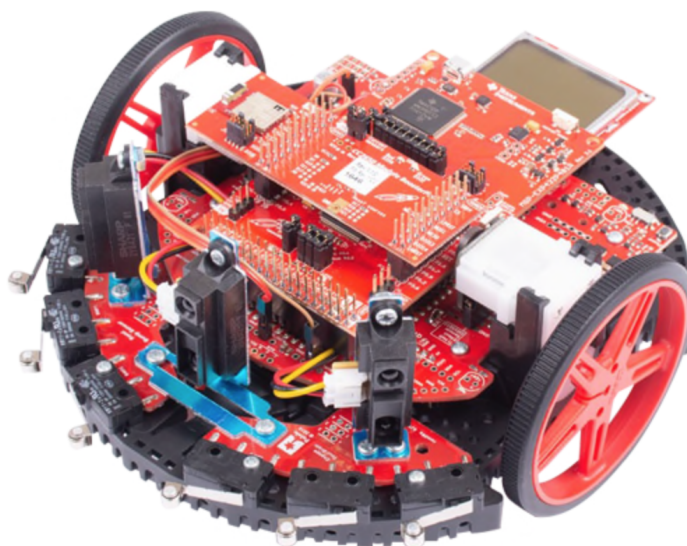
Surgindo como a alternativa a *kits* voltados para a robótica educacional e disponibilizado por umas das maiores fabricantes e distribuidoras de componentes eletrônicos do mundo, O *TI-RSLK* é um *kit* de desenvolvimento que, em comparação ao *LEGO Mindstorms®*, se apresenta mais próximo ao ambiente acadêmico e de engenharia profissional do que de um ambiente completamente recreativo. Enquanto ainda tenta trazer a possibilidade de expansão da criatividade para a mentes dispostas a trabalhar com ele, o *kit* apresentado pela *Texas Instruments* é acompanhado por um manual completo e detalhado que desenvolve fundamentos de sistemas eletrônicos programáveis, por meios de módulos explicativos e uma interface de depuração bastante robusta.

Estruturalmente falando, o *kit* é composto por diferentes módulos, alguns construídos especificamente para realizar as atividades propostas para o conjunto didático, outros reaproveitados de soluções já desenvolvidas pela *Texas* e adaptados para desempenhar funções importantes no pequeno robô. Dentre os módulos inseridos, estão o chassi do robô, a placa-base de conexão dos diferentes periféricos, motores adaptados a rodinhas e também um *kit* de desenvolvimento *SimpleLink® MSP432P410R*, que podem ser visualizados na Figura 4 e na Figura 5.

Debruçando-se mais atentamente à unidade de processamento embarcada, o sistema escolhido para executar as diferentes tarefas propostas para o *TI-RSLK* utiliza um módulo de desenvolvimento poderoso integrado com um processador *ARM® 32-bit* de modelo *Cortex-M4F*, com unidade de ponto flutuante e *DSP* dedicados. Além disso, a unidade de 48 MHz conta com 64 KB de memória *RAM* e 256 KB de armazenamento *Flash*, suporte ao protocolo de criptografia *AES* e integridade *CRC*. Outro ponto forte

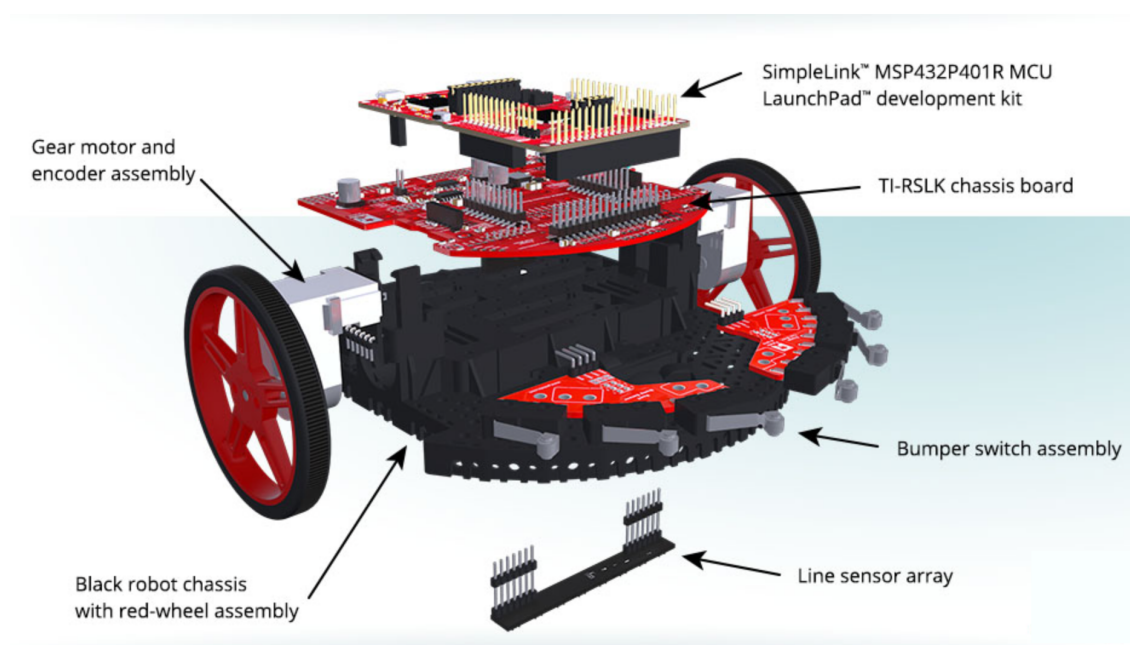
desse núcleo é a presença de um conjunto *ADC* de alta precisão, com 16 *bits* de informação, adaptado a um conjunto de mais de 40 pinos de entrada e saída.

Figura 4 – Kit de desenvolvimento TI-RSLK®



Fonte: <<http://br.mouser.com/>>.

Figura 5 – Módulos associados ao kit de desenvolvimento TI-RSLK®



Fonte: Adaptado de <<http://university.ti.com/programs/RSLK/>>.

Em uma versão mais simplificada do *kit*, apenas as configurações básicas para o funcionamento do conjunto são englobadas, deixando para um segundo *kit*,

mais completo, a adição de módulos de comunicação *Wifi* e *Bluetooth*, sensores de distância, *display LCD* e um sistema de áudio próprio para o brinquedo.

Como citado em seu guia de usuário, o objetivo final do conjunto é o aprendizado, a construção e o teste de um protótipo de robô autônomo, capaz de realizar tarefas simples como percorrer um labirinto, seguir uma linha ou navegar conforme determinados parâmetros configurados, mas, acima de tudo, permitir que a experiência de aquisição de conhecimento seja completa e, ao percorrer todos os módulos, mostrar-se um desafio construtivo e superação das dificuldades.

Analisando o currículo de módulos apresentados pelo *kit* da *Texas Instruments*, é possível notar sua base nos preceitos da própria robótica educacional, servindo como um guia que, ao invés de carregar completamente pela mão, incentiva o usuário a percorrer seu caminho de maneira a compreender na prática todos os fundamentos necessários para o crescimento na área. O conjunto se propõe a tratar de princípios que percorrem diversos tópicos da experimentação científica, indo desde fundamentos de eletricidade, passando por algoritmos matemáticos de processamento de sinais e de dados, aspectos de sistemas embarcados e circuitos eletrônicos e abordagens de lógica para solução de problemas, tudo isso ao mesmo tempo que tenta proporcionar o máximo de interação com o ambiente cotidiano por meio da tecnologia (TEXAS INSTRUMENTS, 2018b).

Discutindo a acessibilidade financeira da solução, o custo final ainda está em um alto nível de investimento. Segundo o site oficial da *Texas Instruments* para o produto, em abril de 2020, o conjunto mais básico do *TI-RSLK* pode ser adquirido pelo valor de US\$ 99,00, enquanto o *kit* que contém as expansões e módulos de comunicação sem fio apresenta um custo de US\$ 199,00. Em uma busca rápida, procurando por revendedores nacionais, não foi encontrada nenhuma loja que disponibilizasse o produto, nem revendedores oficiais ou facilitadores de importação.

2.1.3 *Makeblock®*

No ainda recente mercado de investimento em robótica educacional, não são apenas as grandes empresas de brinquedos ou componentes eletrônicos que investem em soluções para alavancar as técnicas de aprendizado. Um bom exemplo para essa situação encontra-se na *Makeblock®*, empresa fundada em 2013 pelo chinês Jasen Wang e dedicada a soluções para o chamado ensino *STEAM*.

Para fomentar essas vertentes de aprendizado, a *Makeblock®* apresenta diversas linhas de produtos voltadas a diferentes níveis de ensino e conhecimento, permitindo ao usuário encontrar as soluções que mais se encaixam no contexto desejado. Oferecendo desde impressoras 3D e cortadores a laser para customização de superfícies até *kits* completos de componentes eletrônicos, a empresa consegue

atuar como provedora de entretenimento ao mesmo tempo que promove o ensino dos princípios de uso da tecnologia na solução de problemas cotidianos.

Focando no desenvolvimento da programação em fases iniciais, encontram-se os dispositivos *mTiny* (um pequeno robô no formato de um urso panda, que utiliza uma caneta para ler uma sequência de blocos definidas pelo usuário para programar o caminho do robô), *Neuron* (conjunto de blocos eletrônicos, com diferentes tipos de sensores, ligados por contato magnético e com foco no processamento de sinais de estímulo do ambiente) e também o *Codey Rocky* (plataforma que permite a programação de função por meio de blocos digitais e linguagem *Python*), permitindo a introdução dos preceitos de lógica de programação já nas primeiras fases da infância. Na Figura 6, os produtos citados.

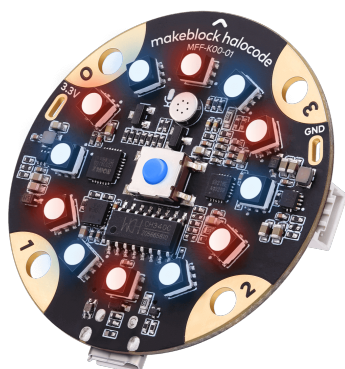
Figura 6 – As soluções *mTiny*(esq.), *Neuron*(centro) e *Codey Rocky*(dir.)



Fonte: Adaptado de <<http://makeblock.com>>.

Partindo para um público de idade um pouco mais avançada e abordando soluções de maior interesse para esse estudo, estão disponíveis conjuntos de *hardware* para a criação de protótipos de robôs automatizados, com diferentes níveis de complexidade e especificações. O primeiro que pode ser citado é o *HaloCode*, uma placa equipada com um *ESP32* embutido e conectada a *LEDs RGB* programáveis, sensores de toque, sensor de movimento, permitindo a conexão wireless por meio de *Wifi* e *Bluetooth* para a programação do conjunto de infinitas maneiras desejadas, tudo isso com o auxílio de uma interface de programação em blocos bastante robusta. Ainda, uma característica interessante é o seu tamanho, de aproximadamente 4,5 cm, e suas inúmeras possibilidades de uso. Na Figura 7, uma imagem de demonstração do produto.

Este é um circuito bastante interessante para a aplicação de programação de sistemas microcontrolados embarcados, abrindo espaço para os conceitos de robótica, entretanto, perdendo a possibilidade de interação com uma estrutura mecânica de partes móveis. É possível que a placa seja incorporada a um conjunto robótico, entretanto, esse contexto não é previsto na comercialização do produto e faz parte da aplicação desejada pelo usuário.

Figura 7 – Placa de desenvolvimento *HaloCode*

Fonte: Adaptado de <<http://makeblock.com>>.

Para focar no desenvolvimento de aplicações em robótica, a *Makeblock* conta com um conjunto de pequenas estruturas voltadas para a integração entre *hardware* e *software*, a linha *mBot*. Passando por diversos níveis de complexidade, a linha apresentada pela *Makeblock®* conta com um modelo de entrada, chamado apenas de *mBot*, sua versão melhorada e intermediária com a alcunha de *mBot Ranger* e um modelo de especificações avançadas chamado *mBot Ultimate 2.0*. Todas as versões do produto são baseadas em soluções que fazem uso de conjuntos de módulos e sensores programáveis capazes de proporcionar a experiência de construção de protótipos criativos, além de desenvolver um fluxo de aprendizado bastante motivador.

Apresentado como o modelo mais básico, o *kit* educacional *mBot* promete um design atrativo e de fácil montagem, indicado até mesmo para crianças, por conta de sua acessibilidade. Em seu funcionamento padrão, o conjunto conta com três modos nativos: 'Desvio de obstáculos', 'Seguidor de linha' e 'Controle manual', sendo este último aprimorado pela compatibilidade com controles remotos. A plataforma *Makeblock®* ainda conta com uma série de softwares de desenvolvimento compatíveis com a linha *mBot*, que permitem desde a adequação ao uso do controle remoto, passando por programação intuitiva por meio de configuração de blocos, chegando até em códigos desenvolvidos em linguagem para a plataforma *Arduino*.

A compatibilidade de desenvolvimento de soluções com linguagem *Arduino* remete diretamente à construção de especificações técnicas do modelo: sua unidade de processamento principal é composta por um *chip ATmega328*, unidade básica dos *kits* de desenvolvimento da família *Arduino UNO®* e que conta com o desempenho de uma plataforma de 8 *bits*, 32 *KB* de memória *Flash* e 2 *KB* de *SRAM*. Sobre os módulos associados, esse *kit* básico é acompanhado de sensor de luminosidade, ultrassônico e um seguidor de linha, além de um receptor IR. Já em termos de conectividade, o *mBot*

tem suporte a conexões *Bluetooth* e pode ser acoplado a outros produtos *Makeblock®*.

Partindo para o modelo intermediário, este apresenta as mesmas funções desempenhadas pelo modelo de entrada, porém, com a adição de alguns novos sensores e modificações na estrutura mecânica permitem outros tipos de implementação ao *kit*. Entre os módulos adicionados estão os sensores de temperatura e de som, e também um giroscópio. Para essa plataforma, o sistema de controle e processamento depende diretamente da placa de desenvolvimento *Arduino Mega2560*, baseado no microcontrolador *ATmega256*, também de 8 *bits* e, neste cenário, alimentado por um *clock* de 16 MHz. Na questão de desempenho, este chip passa a contar com 256 KB de memória *Flash* e 8 KB de *RAM*.

Escalando até a edição mais avançada da solução, o *mBot Ultimate 2.0* possui seu diferencial apenas na maior quantidade de componentes mecânicos que o acompanham. Em comparação ao modelo intermediário, o robô *Ultimate 2.0* possui as mesmas especificações de memória e processamento, utilizando a mesma unidade básica *ATmega2560*. No intuito de apresentar um panorama de evolução, a Figura 8 apresenta os três modelos de robô da linha *mBot*.

Figura 8 – Modelos *mBot*(esq.), *mBot Ranger*(centro) e *mBot Ultimate 2.0*(dir.)



Fonte: Adaptado de <<http://makeblock.com>>.

Como um nível a mais de customização, a linha *mBot* permite ainda a extensão de suas funcionalidades com os chamados *add-ons* compatíveis com as plataformas. Nesses *kits*, são adicionadas peças mecânicas e componentes eletrônicos direcionados a uma construção específica, como a construção de um "pet", robô capaz de emitir sons e responder a estímulos do ambiente, ou mesmo partes suficientes para a construção de um sabre de luz. A adição desses conjuntos requer a compra separada de cada um dos *kits* extras.

Com tantas funcionalidades para explorar, o limite de uso dessa plataforma também é encontrado no investimento necessário para adquiri-la: em consulta realizada em abril de 2020, o modelo *mBot* era oferecido por um valor próximo a US\$ 99,99,

enquanto o *mBot Ranger* era oferecido por US\$ 127,49, chegando ao valor de US\$ 349,99 no modelo *mBot Ultimate 2.0*.

Avaliando o mercado nacional, para o modelo de entrada, um dos valores mais acessíveis, verificado em lojas populares de venda de eletrônicos, girava em torno de R\$ 748,90, aumentando para R\$ 1.497,90, se o desejo fosse de adquirir o modelo *Ranger*. Para o modelo *Ultimate 2.0* não foi encontrada nenhuma alternativa para a aquisição em lojas nacionais, permitindo a especulação de que o preço de investimento não seja compensatório, tomando como base os modelos inferiores que já apresentam valores altos para a realidade enfrentada no mercado eletrônico interno.

Uma discussão interessante é a própria construção do kit citado anteriormente, com especificações técnicas simples, contendo insumos já bastante corriqueiros para o cotidiano de ensino da robótica, como plataformas de desenvolvimento *Arduino* e módulos de sensoriamento vastamente utilizados. Em análise superficial, é possível afirmar que a estrutura de controle e toda a parte que depende apenas de componentes eletrônicos poderia ser replicada de maneira mais barata e mais simplificada. Em contraponto, toda a parte mecânica precisaria de um investimento de tempo e custo para o barateamento de fabricação das peças, dando como exemplo, a criação de modelos 3D que permitam a impressão das peças individualmente, permitindo a replicação do kit em larga escala.

Embora apresente uma ideia bastante alinhada com os propósitos da robótica educacional, além de uma grande variedade de aplicações distintas, os produtos *Makeblock®* não se apresentam como boas alternativas para a mitigação dos problemas de inclusão digital no cenário educacional brasileiro.

2.1.4 *Sphero®*

Outra empresa que apresenta destaque no desenvolvimento de produtos voltados ao aprendizado é a *Sphero®*, empresa americana fundada em 2010 focada no desenvolvimento de kits de robótica acessíveis aos seus usuários. Entre seus produtos famosos encontram-se também robôs derivados de parcerias com o grupo *Disney®* com temáticas relacionadas à saga de filmes *Star Wars®*. Entretanto, para esse estudo, dois são os produtos que despertam interesse de aplicação: o *Sphero BOLT®* e o *Sphero RVR®*.

Focado principalmente no uso da programação como uma forma de interagir com o ambiente, o *Sphero BOLT®* é um robô de carcaça transparente no formato de uma esfera do tamanho aproximado de uma bola de beisebol, adaptado a um pequeno painel de *LEDs* coloridos, completamente programável, acompanhado de uma série de periféricos, como sensores de luminosidade e infravermelho, giroscópio e acelerômetro. Para a programação do robô, o fabricante também proporciona *softwares* compatíveis,

com foco principal em programação por meio de blocos.

A versão mais próxima em comparação aos *kits* apresentados anteriormente, contendo as funcionalidades esperadas para o desenvolvimento da parte de *hardware* no estudo de robótica, é o modelo *Sphero RVR®*, um robô com parte mecânica de movimentação, ou o chassi do carrinho, não-customizável, permitindo a adição de novas estruturas apenas por meio de uma parte reservada no topo do robô. Uma vantagem desta solução é sua compatibilidade para integração de plataformas externas de desenvolvimento, como placas de desenvolvimento *RaspberryPi* e *Arduino*. Para ilustrar, a Figura 9 mostra ambos os modelos citados acima.

Figura 9 – Modelos *Sphero® BOLT*(esq.)e *RVR*(dir.)



Fonte: Adaptado de <<http://www.apple.com/pt>>.

É impossível analisar as soluções sem trazer o comparativo financeiro à tona. Avaliando o mercado americano, no qual o produto está inserindo, a versão *BOLT* pode ser adquirida por meio do site oficial da própria empresa (<www.sphero.com>) pelo valor de aproximadamente US\$ 149,99, conforme verificado em abril de 2020. Para o modelo *RVR*, o valor de venda é próximo a US\$ 249,99. Em contrapartida, buscando por revendedores dentro do mercado nacional, apenas um ponto de venda foi verificado, apresentando somente o robô *BOLT*, por um valor próximo a R\$ 3.200,00.

Ainda, avaliando as vantagens de implementação de cada *kit*, para ambos os modelos, a utilização desses em aplicações mais avançadas não é possível, pelo uso limitado a ambientes mais básicos de ensino, atingindo seu teto de aprendizagem ainda em fases iniciais. Essa negativa é estabelecida ainda sem avaliar o preço, já que esse, mesmo com um cenário que conta com várias funcionalidades inclusas, reflete uma realidade ainda distante.

2.2 Projetos similares

A busca por meios para mitigar as defasagens sofridas pela educação em eletrônica e robótica serve como inspiração para vários projetos de pesquisa e trabalhos de graduação, como forma de oposição à realidade marginalizadora que envolve a educação nas salas de aula brasileiras, em todos os níveis de escolaridade. Não apenas muito importante, essa leva de trabalhos com preocupação socioeconômica é necessária como ferramenta de combate aos diversos percalços cotidianos citados anteriormente.

Dessa maneira, seguem alguns projetos também desenvolvidos com foco na democratização da robótica e da melhoria da difusão tecnológica nos meios educacionais, utilizados como inspiração tanto na condução do estudo teórico quanto no direcionamento das características técnicas envolvidas nesse contexto.

2.2.1 N-Bot

Desenvolvido na Universidade Federal do Rio Grande do Norte, como projeto vinculado a uma tese de doutorado, o robô *N-bot* se encaixa perfeitamente na categoria de projetos que propõe o desenvolvimento de um conjunto com foco em robótica de maneira simples e acessível, preocupado com o custo final do processo. Segundo Aroca (2012), implementar um sistema que consiga aliar intuitividade, custo e funcionalidades é o segredo para uma maior difusão do conhecimento, provocando também a motivação para que outras iniciativas de pesquisa se desdobrem, como forma de democratizar a tecnologia nos espaços escolares.

A maior inovação proposta neste sistema é vista na escolha do método de interação com o protótipo criado, dependente de sinais de áudio para a troca de informações. Por meio de tons do tipo DTMF (*Dual Tone Multi Frequency*) é possível transmitir comandos ao robô de modo a fazê-lo executar uma série de atividades desejadas, substituindo sistemas de comunicação comumente utilizados, como conexões seriais com fios ou mesmo opções *wireless*, como interfaces *Bluetooth*. Para o controle total do robô, qualquer dispositivo capaz de emitir sinais de áudio pode ser utilizado.

A implementação do N-Bot inspirou também alguns fatores do estudo descrito neste documento, principalmente pertinentes às discussões socioeconômicas e estruturação dos requisitos mínimos a se avaliar na interação robô-usuário. É um trabalho bastante curioso, por apresentar um conceito pouco usual nas construções de sistemas robóticos e por fazer uso de princípios complexos encontrados nos estudos sobre eletrônica analógica.

2.2.2 Praxedes

De maneira simplificada, o protótipo denominado *Praxedes*, concebido em um estudo realizado na Universidade Estadual da Paraíba, utiliza uma plataforma bastante conhecida nas esferas de desenvolvimento em robótica: o sistema *Arduino*. Inspirados em conjuntos didáticos comumente comercializados no mercado nacional, Silva e Scherer (2013) montam um robô em formato de carrinho, composto por uma pequena quantidade de componentes e capaz de ser programado para tarefas simples.

Mesmo sendo um sistema pouco complexo, a relevância do projeto apresentado ocorre por incorporar um contexto já conhecido no meio de desenvolvimento hobbista e didático, com plataforma de fácil acesso no mercado e que permite uma customização de componentes.

Ainda, o breve estudo apresenta uma série de custos relacionados, permitindo visualizar os dispêndios econômicos envolvidos. Passando por uma avaliação bruta, o custo dessa alternativa não é necessariamente baixo, entretanto, serve como inspiração para melhorias e aprimoramentos, principalmente com a reavaliação da plataforma *Arduino* utilizada, que possui certo custo associado ao considerar que é uma solução proprietária e que é subutilizada no contexto apresentado.

2.2.3 RoboFácil

Acompanhando a vertente de desenvolvimento de sistemas democráticos para o ensino da robótica, o sistema chamado RoboFácil foi implementado por Miranda, Sampaio e Borges (2010) após uma série de avaliações do custo relacionado aos conjuntos de vasta comercialização naquele período. Assim como os outros projetos apresentados, o RoboFácil teve sua motivação voltada para o desenvolvimento da robótica educacional, reconhecendo que sua aplicação em ambiente escolar incentiva a democratização da tecnologia.

Contando com uma unidade de processamento *CMOS* de baixo consumo, o *hardware* da plataforma idealizada possibilita a conexão de periféricos como motores, sensores de luminosidade e temperatura, além de permitir a utilização de interfaces de entrada e saída, como um *display* e um teclado de cinco botões. Em adição, o sistema conta com conversores analógico-digital (A/D) e digital-analógico (D/A), permitindo a conexão adequada de mais sensores para aferição de sinais do ambiente pelo conjunto.

O sistema preocupa-se também com a interação plataforma-usuário, desenvolvendo um sistema de programação em blocos para reunir as funcionalidades e melhorar a acessibilidade para desenvolvedores que não possuam familiaridade com linguagens de programação tradicionais, focando mais na lógica de programação que no desenvolvimento da sintaxe. Essa abordagem apresenta características bastante interessantes para a aproximação de estudantes de nível fundamental e médio,

incentivando a busca pelo estudo da robótica e eletrônica.

2.3 Possíveis cenários de implementação

Durante o processo de pesquisa e estruturação de desenvolvimento do conjunto didático almejado, várias foram as tecnologias e os métodos avaliados para a implementação do protótipo final, variando entre diferentes tipos de componentes, distintas linguagens de programação ou mesmo estruturas mecânicas peculiares para a montagem do *kit*. Contudo, o critério mais evidente na corrida por um custo mais baixo residiu na análise da plataforma de controle utilizada no robô, parte fundamental no próprio conceito "robótica" e também o item que mais reflete no encarecimento do projeto.

Em uma era em que sistemas microcontrolados atingiram alta popularidade e as tecnologias de semicondutores desempenham um papel bastante importante no progresso tecnológico, a escolha desse item é uma tarefa importante. Por se tratar de uma proposta didática, a pesquisa focou no uso de plataformas de desenvolvimento previamente existentes, ou as chamadas soluções de prateleira *COTS* (*Commercial Off-The-Shelf*), não necessitando de peças de *hardware* extras para atingir seu pleno funcionamento. Nesse contexto, surgiram principalmente três plataformas, citadas nos tópicos a seguir.

2.3.1 Arduino

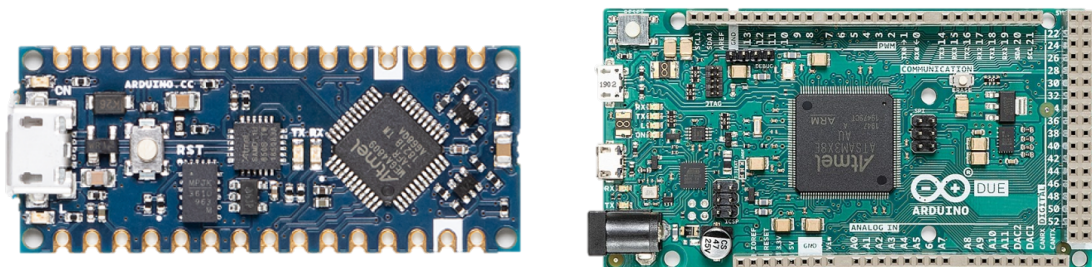
Distribuídas por uma das empresas mais conhecidas no mercado voltado à robótica amadora, principalmente no cenário nacional, as plataformas de desenvolvimento da *Arduino®* alcançam cada vez mais espaço e desenvolvedores fiéis, por aliar um conjunto relativamente barato e com tecnologia suficiente para aplicações de pequeno porte, a uma plataforma de desenvolvimento com linguagem própria de maior acessibilidade ao usuário.

Em termos de software, é possível encontrar em buscas rápidas uma infinidade de projetos para a linguagem utilizada na *IDE* proprietária, capazes de acender um *LED* ou até mesmo criar protótipos de dispositivos biomédicos com suporte a comunicações sem fio. As vantagens evidentes desses conjuntos são intensificadas por conta da própria comunidade de desenvolvedores que, mesmo em caráter amador, desenvolve suas próprias peças de *hardware* adaptáveis, com módulos próprios e os chamados *shields*, que adicionam funcionalidades praticamente ilimitadas aos projetos.

Citando mais especificamente as buscas por soluções que cumpriam os objetivos citados aqui, os modelos sondados e avaliados para uso no desenvolvimento do *kit* didático focaram principalmente em dois modelos da plataforma: o *Arduino*

Nano e o *Arduino Due*. Enquanto o sistema do *Arduino Nano* é mais primitivo e barato, o *Arduino Due* representa um salto na família de placas de desenvolvimento, apresentando um microcontrolador 32 *bits* mais robusto. Para uma melhor visualização, ambos os produtos podem ser verificados na Figura 10.

Figura 10 – Placas de desenvolvimento *Arduino Nano*®(esq.) e *Arduino Due*®(dir.)



Fonte: <<http://store.arduino.cc>>.

A placa do modelo *Arduino Nano*® é um conjunto bastante usual nos kits de robótica de menor custo encontrados no mercado e, assim como os modelos *Uno*® e *Mega 2560*®, é uma plataforma baseada em arquitetura 8 *bits* com memórias *Flash* e *RAM* reduzidas. O modelo *Nano*® difere dos demais apenas pela disponibilidade de pinos de entrada e saída na placa de desenvolvimento. Para se ter uma ideia das capacidades de funcionamento de um *Arduino Nano*, a Tabela 1 evidencia as especificações técnicas do produto, evidenciando os tamanhos reduzidos de armazenamento e processamento do sistema.

Tabela 1 – Especificações técnicas de uma placa *Arduino Nano*®

<i>CPU</i>	<i>ATMega328</i> , 16 MHz
Arquitetura	<i>AVR</i> (8 <i>bits</i>)
<i>Flash</i>	32 KB
<i>SRAM</i>	2 KB
<i>EEPROM</i>	1 KB
<i>GPIO</i>	22 pinos digitais (<i>I/O</i>) 8 pinos analógicos de entrada
<i>PWM</i>	6 saídas

Fonte: <<http://store.arduino.cc>>.

Pensar em plataformas com menor número de funcionalidades é interessante para projetos pequenos por baratear os custos envolvidos, entretanto, pode atuar como limitador na expansão das aplicações dos *kits*, impedindo a adição deliberada de módulos dependentes das interfaces de comunicação proporcionadas pelos periféricos da placa.

Nesse sentido, o *Arduino Due®* pode representar uma boa solução, sendo o primeiro modelo da família a apresentar um núcleo *ARM 32 bits*. O sistema visto nessa plataforma permite a inclusão de rotinas mais rebuscada para gerenciamento de tarefas, comumente implementadas por sistemas operacionais de tempo real (*RTOS*), que conseguem suprir as necessidades para uma vasta quantidade de cenários específicos e complexos. Entretanto, analisando o caráter didático do experimento aqui proposto, pode ser subutilizado e um investimento desnecessário. Para vislumbrar as possibilidades e comparar com as especificações do sistema *Nano*, a Tabela 2 mostra as principais características do conjunto.

Tabela 2 – Especificações técnicas de uma placa *Arduino Due®*

<i>CPU</i>	<i>AT91SAM3X8E</i> , 84 MHz
Arquitetura	<i>ARM®(32 bits)</i>
<i>Flash</i>	512 KB
<i>SRAM</i>	96 KB
<i>GPIO</i>	54 pinos digitais (<i>I/O</i>) 12 pinos analógicos de entrada 2 pinos analógicos de saída (<i>DAC</i>)
<i>PWM</i>	12 saídas

Fonte: <<http://store.arduino.cc>>.

Entendendo a importância do quesito custo, por meio de uma pesquisa rápida em lojas nacionais que comercializam o produto, é possível verificar os preços de varejo das plataformas mencionadas, por meio do conjunto de valores apresentado na Tabela 3.

Tabela 3 – Valores de venda para as placas *Arduino*

Placa	Preço (US\$)	Preço (R\$)
<i>Arduino Nano®</i>	20,70	29,00 - 39,90
<i>Arduino Uno®</i>	23,00	59,90 - 62,90
<i>Arduino Mega 2560®</i>	40,30	109,90 - 119,90
<i>Arduino Due®</i>	40,30	119,90 - 189,90

Fonte: <<http://store.arduino.cc>>.

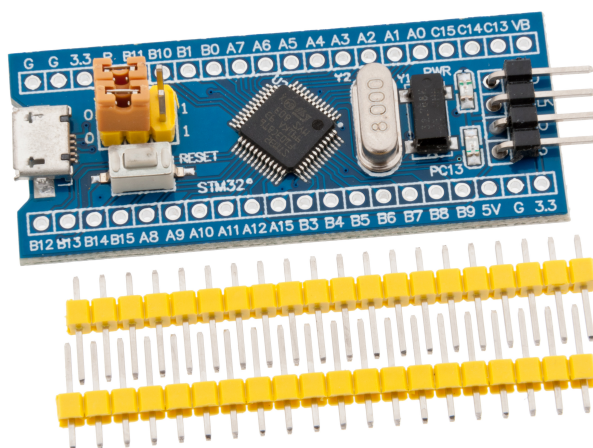
Outro ponto interessante a ser discutido é a compatibilidade das linguagens de programação nos *kits*, que são completamente adaptadas para desenvolvimento em linguagem *C/C++*, bastante usuais nos ambientes de criação de sistemas embarcados. O sistema *Arduino* ainda conta com *IDE* adaptada e biblioteca própria para abstração direta de configurações direcionadas às placas proprietárias, trazendo a usuários iniciantes uma menor preocupação com o conhecimento sintático sobre a linguagem e uma maior liberdade para focar nas funcionalidades da solução desejada.

2.3.2 Bluepill STM32F103C8TX

Atuando como uma das maiores distribuidoras do mundo de componentes baseados em semicondutores, a *ST Microelectronics* também está completamente inserida no mercado de plataformas microcontroladoras e disponibiliza uma enorme família de circuitos, com especificações técnicas complexas e um custo bastante acessível. Dentre os componentes dessa linha, encontram-se os modelos *STM32F103*, dispositivos baseados em arquitetura *ARM Cortex-M3* e com frequência de operação que pode chegar a 84 MHz. Além disso, fazem parte da família de produtos *STM32F103* modelos que possuem memória de armazenamento em *Flash* de tamanhos variados, desde 16 *kB* a 1 *MB*, expandindo os limites da quantidade de informação inserida e impactando na complexidade dos produtos.

Direcionando os esforços para este estudo, o modelo que mais se enquadra na proposta de desenvolvimento em baixo custo é o *STM32F103C8TX*. O principal diferencial desta solução é apresentar as qualidades da arquitetura *ARM*, tanto em questão de desenvolvimento quanto na robustez de execução das tarefas. De forma comparativa, o *STM32F103C8TX* está no patamar do *Arduino Due®*, também com a arquitetura 32 *bits*, entretanto, com séria vantagem competitiva nos custos. Uma imagem ilustrativa de uma placa de desenvolvimento baseada no *STM32F103C8TX* pode ser vista na Figura 11.

Figura 11 – Placa de desenvolvimento do modelo *STM32F103C8TX*



Fonte: <<http://www.electrokit.com/en>>.

Focando nos aspectos técnicos, o microcontrolador possui uma série de características interessantes, que podem ser visualizados na Tabela 4.

Tabela 4 – Especificações técnicas de um microcontrolador *STM32F103C8TX*

<i>CPU</i>	<i>Cortex-M3, 72 MHz</i>
Arquitetura	<i>ARM®(32 bits)</i>
<i>Flash</i>	64 ou 128 KB
<i>SRAM</i>	20 KB
<i>GPIO</i>	37 pinos digitais (I/O)
Interfaces compatíveis	<i>7 timers</i> até 3 <i>USARTs</i> até 2 <i>SPIs</i> até 2 interfaces I ² C interface <i>CAN</i> interface <i>USB 2.0</i>

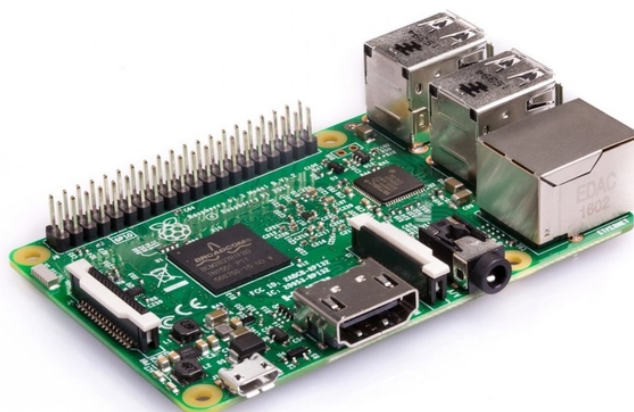
Fonte: <<https://www.st.com/>>.

Novamente, de forma comparativa, embora não se iguale às capacidades de processamento e armazenamento encontradas na plataforma *Arduino Due*, por exemplo, o microcontrolador *STM32F103C8TX* supera as características apresentadas pelos modelos mais simples, com maior velocidade e uma arquitetura mais complexa. Ainda, o custo é também um fator significativo, competindo, por exemplo, com o próprio *Arduino Nano®*. Buscando em lojas brasileiras, o *STM32F103C8TX* pode ser encontrado por um custo a partir de R\$ 24,00 podendo variar até valores próximos a R\$ 39,00, ainda acessível.

Para o desenvolvimento, a *ST Microelectronics* fornece uma completa biblioteca desenvolvida em linguagem C e responsável por configurar as diversas camadas necessárias para aplicações diversas, podendo ser voltadas às interrupções das interfaces ou gerenciadas por sistemas operacionais dedicados. Com esses requisitos, é possível recomendar o uso deste dispositivo na construção do *kit* didático, ao conseguir equilibrar funcionalidade e baixo custo.

2.3.3 *Raspberry Pi*

Partindo para conjuntos de desenvolvimento mais complexos, uma plataforma que se destaca por ser completa e autossuficiente, literalmente um computador de uma placa só, é o *kit Raspberry Pi®*. Depois de uma série de modelos lançados pela Fundação *Raspberry Pi*, os produtos *Raspberry Pi®* fazem parte de diversas implementações com foco em robótica. Na Figura 12, é possível vislumbrar a construção de um dos modelos mais recentes comercializados.

Figura 12 – Placa de desenvolvimento do modelo *Raspberry Pi 3 Model B*

Fonte: <<http://www.raspberrypi.org>>.

Citando suas funcionalidades, algumas especificações técnicas das diferentes versões da plataforma *Raspberry Pi* podem ser conferidas na Tabela 5.

Tabela 5 – Especificações técnicas de diferentes modelos de *RaspberryPi*

	Pi Zero W	Pi 3 B+	Pi 4 B
<i>SoC</i>	<i>Broadcom BCM2835</i>	<i>Broadcom BCM2837</i>	<i>Broadcom BCM2711</i>
<i>CPU</i>	1x <i>ARM1176JZFS</i>	4x <i>ARM Cortex-A53</i> 64 bits	4x <i>ARM Cortex-A72</i> 64 bits
<i>Clock</i>	1 GHz	1.2 GHz	1.5 GHz
<i>GPU</i>	<i>Broadcom VideoCore IV</i>	<i>Broadcom VideoCore IV</i>	<i>Broadcom VideoCore IV</i>
<i>RAM</i>	512 MB	1 GB	2, 4 ou 8 GB
Armazenamento	Cartão <i>MicroSD</i>	Cartão <i>MicroSD</i>	Cartão <i>MicroSD</i>
<i>GPIO</i>	40 pinos	40 pinos	40 pinos
<i>Bluetooth</i>	4.1(<i>BLE</i>)	4.2(<i>BLE</i>)	5.0(<i>BLE</i>)
<i>Wireless</i>	802.11n	802.11n 2.4GHz/5.0GHz	802.11ac 2.4GHz/5.0GHz
Interfaces diversas	mini- <i>HDMI</i> mini- <i>USB</i> (comunicação) micro- <i>USB</i> (alimentação)	<i>HDMI</i> Conector <i>P2</i> - 3,5mm 4 portas <i>USB 2.0</i> Porta <i>Ethernet</i>	4x <i>HDMI</i> Conector <i>P2</i> 3,5mm 2x <i>USB 2.0</i> 2x <i>USB 3.0</i>

Fonte: <<http://www.raspberrypi.org>>.

Um aspecto interessante do conjunto é a ausência de uma memória *flash* fixa, sendo substituída por uma entrada de cartão *microSD*, permitindo uma custo-

mização dessa memória não-volátil conforme a necessidade da própria aplicação instalada. Sendo esse um computador portátil, ao inserir um cartão, é possível instalar nele uma distribuição de um sistema operacional *Linux*, permitindo gerenciar todas essas funcionalidades, seja por meio de interface gráfica ou linhas de comando. Para o sistema *Raspberry* está disponível uma distribuição própria de sistema operacional, denominada *Raspbian*, adaptação da distribuição *Debian* e com principal vantagem residindo em baixo consumo de armazenamento.

Ainda, pensando em um robô com formato de carrinho, é natural idealizar um sistema livre de cabos que o prendam em um único espaço e limitem sua circulação. Nesse sentido, uma proposta interessante é o uso de um acesso remoto da interface de linha de comando do sistema, por meio do protocolo de conexão remota segura chamado *SSH(Secure Shell)*. Por ter suporte à conexão *Wifi*, o sistema pode ser acessado remotamente e seus arquivos internos editados de maneira fácil.

Em complemento, os sistemas operacionais *Linux* possuem ótima compatibilidade com a linguagem de programação *Python*, uma linguagem de alto nível e com uma infinidade de módulos e bibliotecas disponíveis para adição, além de extensa documentação.

Assim como avaliado no contexto das outras plataformas discutidas anteriormente, o quesito custo também é impactante. Para uma avaliação prática, a Tabela 6 apresenta alguns valores de venda para estes dispositivos.

Tabela 6 – Valores de venda para as placas *RaspberryPi*

Placa	Preço (US\$)	Preço (R\$)
<i>Pi Zero W</i>	10,00	159,90
<i>Pi 3 Model B+</i>	35,00	439,90
<i>Pi 4 Model B (2 GB RAM)</i>	35,00	439,90
<i>Pi 4 Model B (4 GB RAM)</i>	55,00	599,00

Fonte: <<http://www.raspberrypi.org>>.

Diferente das placas de desenvolvimento da *Arduino*, os produtos com selo *Raspberry Pi* sofrem demais com as taxas de conversão e chegam ao país com valores fora da faixa do baixo custo. Entretanto, a escolha dessa solução pode ser feita por apresentar um equipamento útil para diferentes propósitos, muito além da construção de um robô didático, trazendo a possibilidade de um computador de placa única, ainda assim mais acessível que as versões mais baratas de computadores pessoais comuns. O pensamento se estabelece em depender de uma plataforma bastante conhecida e de alta possibilidade de reutilização.

3 PROPOSTA DE PROJETO

Reunindo todas as inspirações em alternativas comerciais e informações previamente apresentadas, este estudo busca a apresentação de um conjunto de maneiras simples de alcançar resultados muito parecidos com as referências descritas, focando em viabilidade de custo e acesso aos insumos tecnológicos necessários.

Espelhando-se nas características técnicas descritas no capítulo 2, é importante estabelecer um escopo de desenvolvimento, estabelecendo as metas dessa alternativa para os convencionais *kits* de robótica. Os tópicos a seguir descrevem brevemente as intenções de cada uma das áreas de abordagem para o projeto, com propostas de confecção de plataformas de *hardware*, *firmware* e *software*.

O público-alvo a que se destina este desenvolvimento são estudantes que realizarão a montagem e adequação do *kit* didático, deixando a plataforma preparada para o desenvolvimento posterior de atividades voltadas ao ensino e ao desenvolvimento cognitivo. Não está prevista a criação de roteiros que abordem especificamente o aprendizado de assuntos práticos ou ementas de salas de aula, focando apenas nas decisões contrutivas da plataforma.

3.1 Descrição geral

Em termos gerais, o *kit* didático deve ser uma plataforma simples e de baixo custo, com o máximo de funcionalidades possíveis. Considerando uma escala reduzida, o resultado do desenvolvimento pretende competir em termos de funcionalidade com as plataformas anteriormente citadas, apresentando os mesmos contextos de ferramentas educacionais, entretanto, focando na fácil construção e acessibilidade.

Além disso, aliado ao desenvolvimento, o foco do estudo é manter uma documentação completa e acessível das ferramentas e processos empregados, utilizando plataformas de desenvolvimento gratuitas, comum ao estudo da robótica livre e que torna a reprodutibilidade do conjunto facilitada, permitindo a qualquer interessado que tenha seu próprio *kit*. Para isso, serão discutidas as escolhas relacionadas a cada trecho documentado, permitindo que alterações futuras sejam feitas de maneira planejada e a personalização de cada contexto seja permitida. Os tópicos a seguir tratam das especificações técnicas desejadas para cada âmbito de trabalho.

3.2 Especificações

3.2.1 Estrutura mecânica

O desenvolvimento da estrutura mecânica de um robô é importante por sua responsabilidade no processo de identificação do usuário com o artefato de aprendizado

pois, como propõe César (2020), "o modo como o ser humano percebe e interpreta a si mesmo a partir da interação subjetiva com o artefato é sua maneira pessoal de vivenciar as experiências de aprendizagem e desenvolver a 'cognição'". Construir plataformas que se assemelham a estruturas cotidianas permite uma maior integração do objeto de estudo no ambiente em que está incluído, estimulando a criatividade do aluno na resolução de problemas que ultrapassam o escopo estritamente didático.

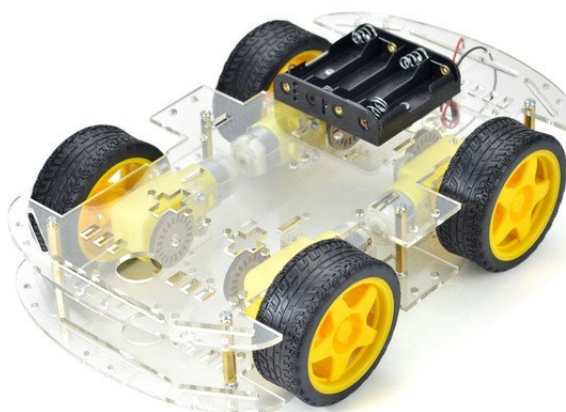
Pensando nessa familiaridade, o uso de formas e estruturas que se assemelham a sistemas vistos na natureza, como o corpo de organismos vivos, despertou nos estudos em robótica o interesse por sistemas conhecidos como "robôs moles", apresentados por Campo (2020) como "dispositivos construídos com diferentes tipos de materiais, incorporando sensores e atuadores biologicamente inspirados para uma operação mais segura e complacente". O uso de robôs moles é visto como o futuro por conta de sua natural evolução aos clássicos modelos de robôs rígidos, apresentando maior adaptabilidade na interação com o ambiente e maior integração com a realidade.

Entretanto, os maiores obstáculos no uso dessas características estão no projeto de dispositivos deste tipo, já que o aumento de graus de liberdade eleva a complexidade no planejamento, pela presença de trajetórias imprecisas e um sensoriamento de posição complexo. Adicionada a isso, a obtenção de materiais adequados para a construção de dispositivos desse molde pode ser um fator inviável financeiramente (CAMPO, 2020).

Dessa forma, o kit didático pensado para esse estudo ainda mantém a idealização de um robô rígido, pela maior previsibilidade de projeto e disponibilidade de soluções já comercializadas para este fim. Assumindo a forma de um carrinho automatizado com quatro motores, o sistema fará uso de um modelo comercial de chassi, feito de acrílico, com dimensões aproximadas de 15x25 cm (LxC), e escolhido pela facilidade de acesso desse insumo em mercado nacional. A estrutura conta com duas chapas paralelas e um espaço entre elas, permitindo o armazenamento de baterias para a alimentação do sistema e furos para a fixação dos quatro eixos dos motores.

A superfície superior da placa será utilizada para a distribuição dos componentes eletrônicos necessários às implementações, permitindo o posicionamento de uma placa de circuito impresso de dimensões máximas de 10x17 cm (LxC). Já as rodas selecionadas para esse carrinho, possuem aro de plástico com diâmetro aproximado de 5 cm envolto em borracha, em formato típico de carrinho de brinquedo. A Figura 13 mostra o chassi adquirido para a utilização no projeto.

Para suportar as condições de funcionamento dos motores, deverá ser verificado o uso de baterias adequadas, com valores de tensões e correntes suficientes. Os motores adquiridos funcionam com tensões entre 3 e 6 V e, mesmo sem gerar

Figura 13 – Chassi utilizado como base para o conjunto didático

Fonte: <<http://www.eletrogate.com>>.

torque, exigem uma corrente mínima de aproximadamente 150 mA para a menor tensão, podendo aumentar até 200 mA para uma tensão máxima aplicada sobre seus terminais. As dimensões das baterias também precisam ser previamente verificadas para que sejam acomodadas nas mesmas dimensões disponíveis para a placa, posicionada no espaço entre as chapas do chassi, com limitação de altura máxima em 3 cm.

Motores com essas características são comumente encontrados sem a especificação de um fabricante, apresentando dificuldade de acesso a folha de dados. De todo modo, um dos modelos frequentemente encontrados, já acompanhados de estruturas de adaptação de seu eixo, possui características mecânicas semelhantes ao verificado na Figura 14.

Figura 14 – Motor de corrente contínua utilizado no projeto

Fonte: <www.eletrogate.com>.

3.2.2 Hardware

Seguindo a ideia da robótica livre, apresentada por César (2020), é interessante o projeto customizado e a confecção de uma placa de circuito impresso própria, com as seguintes proposições: o esquemático e o *layout* da placa serão desenvolvidos por meio de *softwares* de desenho apropriados, com distribuição livre, respeitando as dimensões máximas ditadas pelo chassi escolhido. Além disso, pensando no processo artesanal de confecção, a placa terá melhor desempenho se for de face simples e com distanciamento entre trilhas suficiente para reduzir curtos provenientes de imprecisão na corrosão do cobre.

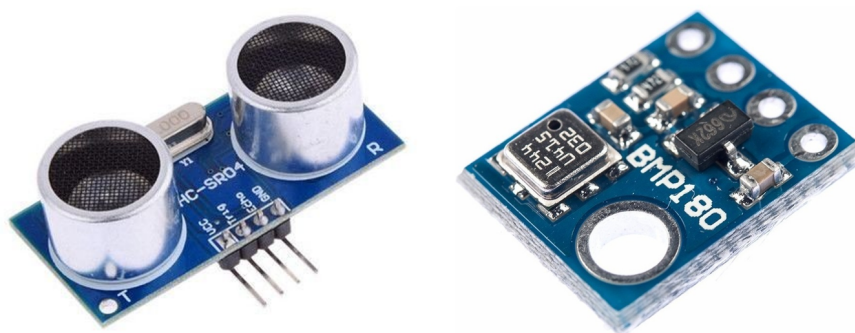
Um dos principais circuitos a ser desenvolvido deve ser o de acionamento dos motores, primando por topologias simples. Para a bateria, um sistema de medição da tensão entre os polos deve ser desenvolvido de modo a sinalizar ao usuário a necessidade de recarga.

Tomando como inspiração as soluções comerciais avaliadas anteriormente, é possível perceber que a inclusão de diferentes módulos e sensores ao conjunto também é imprescindível. Nesse processo, foram selecionados alguns dos sensores vistos nestes *kits*, além de avaliada a disponibilidade imediata dos insumos, resultando na seleção dos seguintes itens: sensor ultrassônico de distância, sensor de temperatura e pressão, acelerômetro e giroscópio, sensores ópticos e um módulo de comunicação *Bluetooth*.

O popular módulo de sensor ultrassônico *HC-SR04* é responsável pela medição de diferentes distâncias com a emissão de um sinal ultrassônico e a posterior medição do tempo de reflexão de ondas mecânicas. É um módulo de fácil utilização e comumente empregado em atividades com navegação, por ser ótimo para a identificação de barreiras. Este dispositivo deve ser alimentado com uma tensão de 5 V e é capaz de medir distâncias desde 2 cm até 4 m.

O módulo BMP180 é um sensor barométrico de baixo consumo, além de bastante compacto. Juntamente com informações de pressão, esse módulo é capaz de informar valores de temperatura entre -40 e +85 °C. O módulo pode ser alimentado com tensões entre 1,8 e 3,6 V e, para a comunicação e transmissão de dados, conta com uma interface I²C. De maneira ilustrativa, a Figura 15 apresenta os dois sensores mencionados.

Figura 15 – Módulo sensor ultrassônico *HC-SR04* (esq.) e módulo sensor barométrico *BMP180* (dir.)

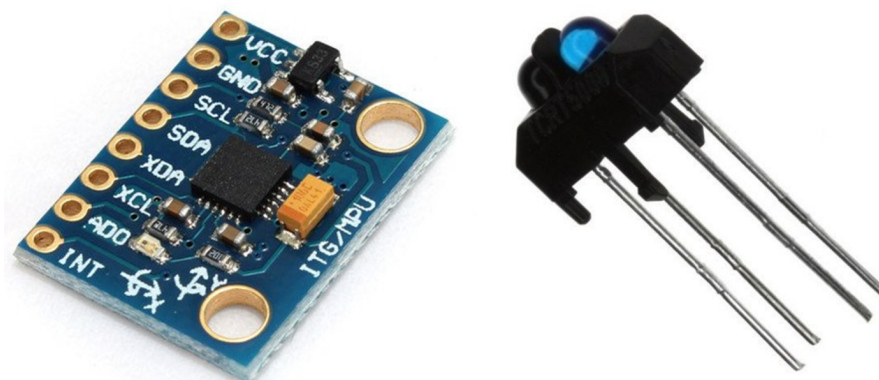


Fonte: <<http://www.eletrogate.com>>.

Com a popularização de smartphones e outros equipamentos portáteis, aumentou também o número de possibilidades de uso dos sensores chamados acelerômetro e giroscópio, capazes de verificar posição relativa espacial e deslocamento. Por tratar-se de um sistema com mobilidade, a adição do módulo GY-521 é bastante interessante e permite a leitura das informações nos eixos dos sensores de maneira direta. Alimentado com tensões entre 3 e 5 V, o módulo conta com ADC interno e transmite suas informações por meio de uma interface I²C, podendo ser acoplado paralelamente ao barramento utilizado para o *BMP180*.

Ainda pensando no contexto de uma plataforma com mobilidade, um sensor óptico do modelo *TCRT5000* apresenta uma nova possibilidade de interação do sistema com o ambiente, podendo identificar variações de cor. Comumente utilizado para implementação de um modo de funcionamento chamado "seguidor de linha", esse sensor permite o reconhecimento de um caminho definido a ser percorrido pelo robô, fazendo com que percorra um circuito de maneira autônoma. Comercialmente, é possível encontrar um módulo de leitura pronto para o sensor *TCRT5000*, entretanto, por ter um circuito simples, será feita a utilização do sensor isolado em conjunto com componentes discretos para a captura das variações de luminosidade. Na Figura 16, é evidenciado o módulo de posicionamento e do sensor óptico que devem ser agregados ao *kit*.

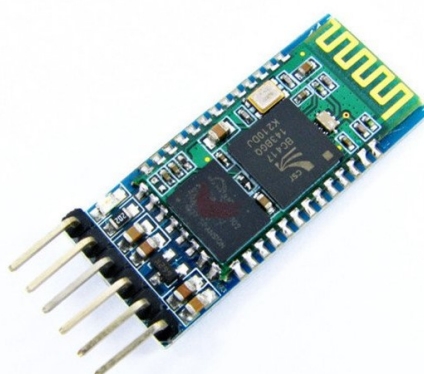
Figura 16 – Módulo sensor giroscópio e acelerômetro GY-521 (esq.) e sensor reflexivo TCRT5000 (dir.)



Fonte: <<http://www.eletrogate.com>>.

Por fim, além de adquirir essas informações, é importante fazer uso de uma interface de usuário que suporte o envio e a recepção de eventos ao robô, permitindo controlar o sistema externamente. Para essa finalidade, pode ser utilizado um módulo bastante comum em implementações de robótica, o *HC-05*, baseado na conexão de dispositivos via *Bluetooth*. O sistema suporta tensões entre 3 e 6 V e faz uso de uma interface serial do tipo *UART* para a troca de informações com o microcontrolador. A Figura 17 ilustra a aparência física do módulo.

Figura 17 – Módulo de comunicação *Bluetooth HC-05*



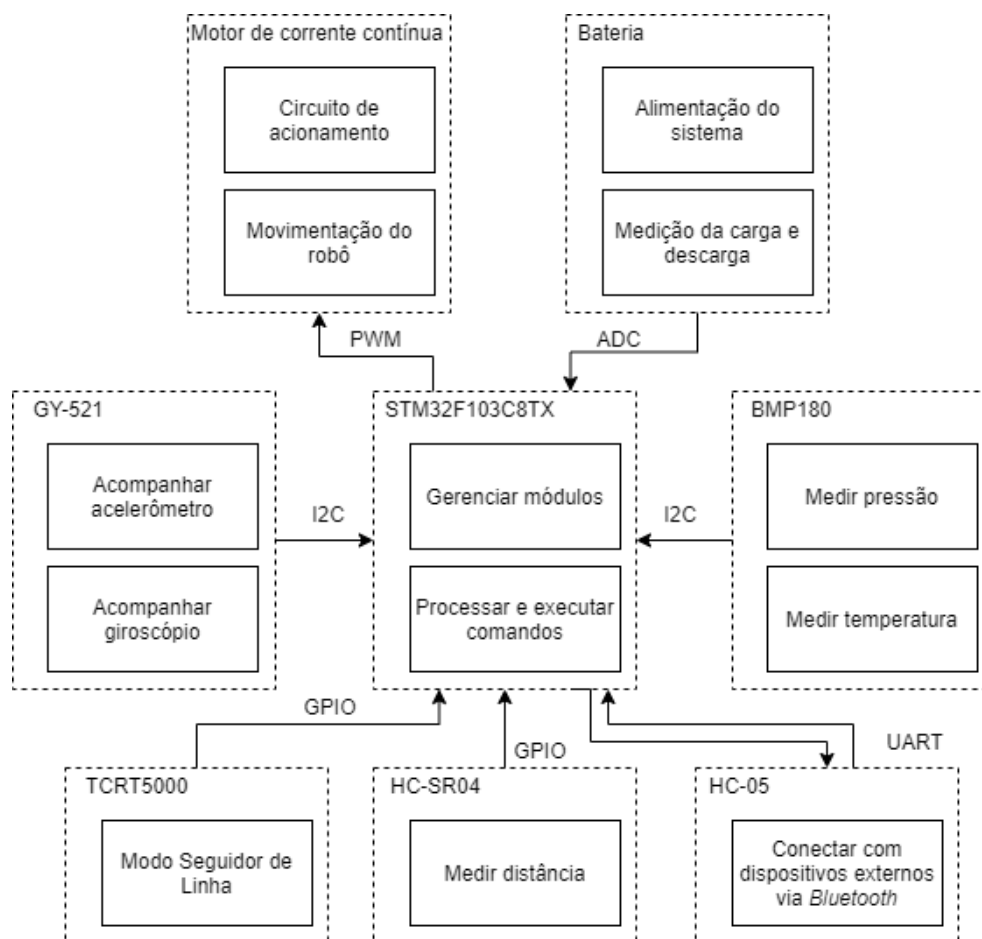
Fonte: <<http://www.eletrogate.com>>.

Retomando os questionamentos levantados na seção 2.3, é também de grande importância a seleção de uma plataforma de controle adequada à complexidade esperada ao kit, pois esta servirá como o "cérebro" do sistema, realizando as tarefas de gerenciamento das funcionalidades propostas. Tomando os dispositivos pesquisados, o mais apropriado para os requisitos do projeto foi a solução apresentada na subseção 2.3.2, a *Bluepill STM32F103TX*, com capacidade de processamento

bastante apropriada ao escopo definido, facilidade de acesso no mercado nacional e um custo reduzido. Analisando as especificações técnicas, o sistema é suficiente com suas interfaces de comunicação ao apresentar ao menos as interfaces I^2C e $UART$, necessárias para a comunicação com alguns dos módulos.

A reunião de todos esses subconjuntos resulta em um sistema integrado programável, capaz de adquirir informações do ambiente externo e tomar decisões para a interferência sobre o mesmo. Na Figura 18, está representado um diagrama de blocos que relaciona os módulos apresentados, servindo como base para estruturar a aplicação tanto na organização física dos componentes quanto na criação do programa que fará o gerenciamento do protótipo.

Figura 18 – Diagrama de blocos com as funções básicas previstas para o sistema



Fonte: (Autoria Própria, 2020).

3.2.3 Firmware e Software

Ao estabelecer definitivamente a plataforma de desenvolvimento, é necessário selecionar a linguagem de programação a se utilizar no desenvolvimento dos processos de automação do conjunto. Embora seja possível converter qualquer outra

linguagem de programação para linguagem compreensível pelo processador com seu banco de instruções em *assembly*, a distribuidora dos sistemas *STM32*, *ST Microelectronics®*, possui em seu site oficial uma série de arquivos de documentação e recomendações para o emprego de *softwares* proprietários para a programação do *kit*. Toda essa documentação é voltada para as linguagens de programação *C/C++* e, portanto, esta foi a linguagem selecionada para direcionar o processo de desenvolvimento.

Falando mais especificamente das plataformas disponibilizadas para a programação do microcontrolador, duas delas serão utilizadas no desenvolvimento desse projeto: o *software* de gerenciamento de projetos *STM32CubeMX* e a interface de desenvolvimento *Atollic TRUEStudio*, ambos oficialmente sugeridos pela distribuidora como ferramentas de interação com o sistema. É possível também utilizar a interface de programação *Arduino®* e seus códigos em linguagem específica para compilar programas próprios para o *STM32*, sendo uma alternativa também gratuita às plataformas sugeridas pela empresa *ST*.

3.2.4 Aplicativo para Dispositivos Móveis

Pensando na utilização do módulo do *Bluetooth* para a interação externa com o sistema do *kit*, é interessante a busca por aplicativos para dispositivos móveis dotados de interface sem fio, ou mesmo a criação de um aplicativo simplificado que permita a troca de informações e foque nos critérios de usabilidade do sistema proposto.

Para qualquer umas das situações, com aplicativo já existente ou com a criação de uma nova ferramenta, é importante a busca pelo uso de plataformas acessíveis que possam ser reproduzidas com a mesma facilidade que o *kit* didático, além de serem compatíveis com a ideia de robótica livre discutida anteriormente.

Como ferramentas principais, o aplicativo deve compreender algumas funcionalidades básicas: ter a possibilidade de enviar comandos de controle para a plataforma, permitindo acionar o sistema de diferentes formas, controlando os motores e fazendo requisições de status; em contraponto, o aplicativo deve estar disponível para receber e exibir as informações adquiridas pelos sensores acoplados, permitindo ao usuário que veja o pleno funcionamento do sistema.

4 DESENVOLVIMENTO DO SISTEMA PROPOSTO

4.1 Considerações Gerais

Os tópicos a seguir explicitam os processos e estudos desenvolvidos para a construção de um kit didático, divididos em três etapas, cobrindo as implementações necessárias e as decisões intermediárias que moldaram o resultado final do projeto, explorando de maneira simplificada as situações para permitir a posterior personalização dos módulos, de acordo com as aplicações pretendidas.

4.2 Desenvolvimento do *Hardware* Eletrônico para o Sistema Proposto

Em ordem cronológica, a primeira etapa desenvolvida para o projeto está diretamente relacionada com a parte de *hardware* e os componentes eletrônicos pertencentes aos subcircuitos do sistema, idealizando um modelo que seja capaz de compreender e conectar adequadamente todas as partes, resultando na construção de uma Placa de Circuito Impresso (PCI). Embora seja possível realizar o desenho manual dessas placas, mesmo na escolha de um processo artesanal de fabricação, costuma-se fazer uso de *softwares* de projeto e desenho que facilitam as conexões e permitem um trabalho mais preciso e minucioso, inclusive para *layouts* direcionados à confecção industrial.

São diversos os exemplares de *softwares* que podem ser utilizados no projeto de uma placa, entretanto, por ter foco na utilização de ferramentas de baixo custo e fácil acesso, como sugere o conceito de robótica livre, para este estudo foi escolhida a ferramenta computacional *KiCad®*, em sua versão 5.0.0, gratuita e de código aberto, compatível com diversos sistemas operacionais e com uma extensa biblioteca de símbolos disponível, tanto para o esquemático quanto seus encapsulamentos (*footprints*) associados. Ainda, permite a criação de novos modelos, importação de arquivos e criação de documentos auxiliares comuns em um projeto, como listas de materiais e modelos próprios para a confecção industrial.

4.2.1 Definições do Diagrama Esquemático do Sistema Proposto

O desenho de um esquemático consiste basicamente na identificação dos componentes necessários ao circuito e na garantia de que as conexões entre eles são estabelecidas de maneira correta. As decisões tomadas nessa etapa do projeto refletem diretamente no funcionamento do circuito, já que este funciona como espelho tanto para a distribuição correta dos conjuntos no desenho de *layout* quanto para a posterior identificação dos componentes no momento da montagem.

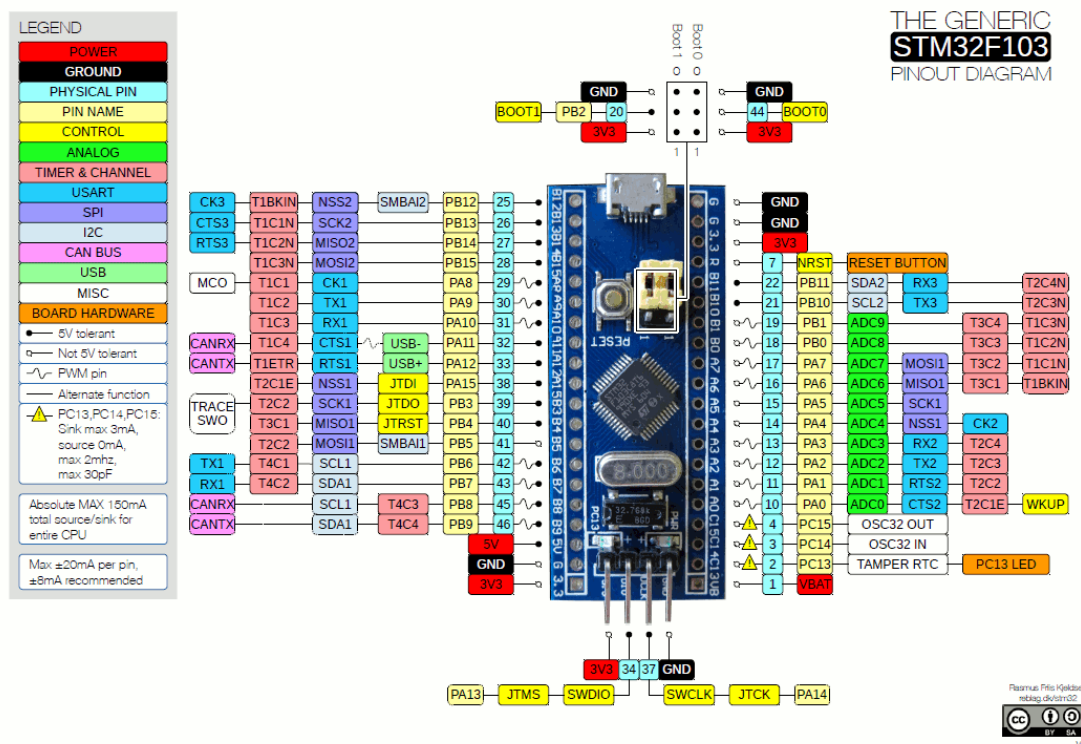
Ainda, é importante citar que, para iniciar o desenvolvimento deste projeto, algumas documentações prévias, de trabalhos anteriores desenvolvidos por Starke (c. 2020), foram disponibilizadas e serviram como base para a construção das funcionalidades de *hardware* esperadas para o robô. Essa documentação foi bastante importante, pois testes preliminares com as implementações propostas comprovaram o funcionamento adequado da solução, que apresenta comportamento próximo ao esperado como resultado para este estudo. A utilização desta fonte de informações se limitou apenas à reprodução de algumas das topologias utilizadas, assim como a seleção de modelos de componentes para estas, analisando a viabilidade. A construção da placa de circuito impresso não foi reproduzida, sendo completamente refeita na proposta deste projeto, assim como o programa de *software*, que foi desenvolvido integralmente.

4.2.1.1 O microcontrolador

Por ilustrar um sistema com gerenciamento centralizado em um microcontrolador, o primeiro dispositivo a ser posicionado no diagrama foi a placa *Bluepill STM32F103C8TX*. A interação dos outros circuitos com este componente depende de uma verificação das características de cada pino, conferindo qual interface está vinculada a cada entrada e também os limites elétricos para cada porta. Para facilitar esse processo, é importante conhecer as configurações básicas de *layout* do módulo que compõe a placa do microcontrolador, normalmente descrito em folhas de dados (ST MICROELECTRONICS, 2015). Na Figura 19 é possível visualizar um diagrama com a descrição funcional de todos os pinos disponíveis.

Embora o microcontrolador tenha sua operação interna dependente de um valor de tensão entre +2,0 V e +3,6 V, a plataforma *Bluepill* possui em sua placa um regulador de tensão que permite, por meio do pino designado, fornecer uma entrada com valor +5,0 V (limite em +5,5 V), usual para circuitos eletrônicos, fazendo com que, além de alimentar corretamente o microcontrolador, todos os pinos marcados com o valor +3,3 V sirvam como ponto de alimentação a outros circuitos periféricos a ele. Em termos de potência, com dados baseados em folhas de dados, o dispositivo regulador comumente utilizado nesta plataforma é um *Low-Drop Output (LDO)* de modelo *TX6211B*, o que permite o fornecimento de valores máximos de corrente próximos a 400 mA (SHANGAI TX ELECTRONICS, 2015).

Nos outros pinos demarcados com retângulos pretos, também é possível identificar conexões com a referência do sistema, apresentando valor de tensão nulo (0 V), todas elas conectadas internamente e disponíveis para a utilização como pontos de distribuição para outros circuitos.

Figura 19 – Relação dos pinos de conexão à plataforma *Bluepill STM32F103*

robô. Como mencionado nas especificações do projeto, estes motores podem ser alimentados com os 5 V de tensão estabelecidos e, quando em operação, cada um deles necessita de uma corrente de aproximadamente 200 mA circulando entre seus terminais, resultando em um consumo de no mínimo 800 mA, sem considerar qualquer um dos outros componentes do *kit*. Como os módulos de sensores e outros elementos dos circuitos do motor não apresentam consumo muito elevado, o consumo total não será aumentado de forma muito significativa ao considerar os outros conjuntos. Para garantir uma tolerância adequada de fornecimento, foi selecionado um modelo de bateria com capacidade de 1,800 mAh, capaz de fornecer no período uma hora aproximadamente o dobro da corrente mínima necessária. Grande parte das baterias do tipo *Li-Po* que possuem essa característica, são encontradas em comercialização com um encapsulamento mecânico padronizado, que pode ser visualizado na Figura 20.

Figura 20 – Bateria *Li-Po* para alimentação do sistema



Fonte: <www.americanas.com>.

A vida útil de uma bateria *Li-Po* depende de alguns cuidados básicos, sendo um destes a garantia de ela não atinja valores de tensão abaixo de um limiar crítico, tornando necessário o acompanhamento progressivo dos níveis de carga. É importante lembrar que, mesmo conectando duas células em série e obtendo um valor de tensão único, a descarga delas pode ocorrer de maneira desproporcional, obrigando o acompanhamento individual da tensão de ambas. Para isso, o primeiro ponto de leitura será na ligação entre as duas células, que pode atingir valores de até aproximadamente +4,2 V; o segundo ponto de leitura será feito no polo positivo do barramento, podendo atingir valores de até +8,4 V.

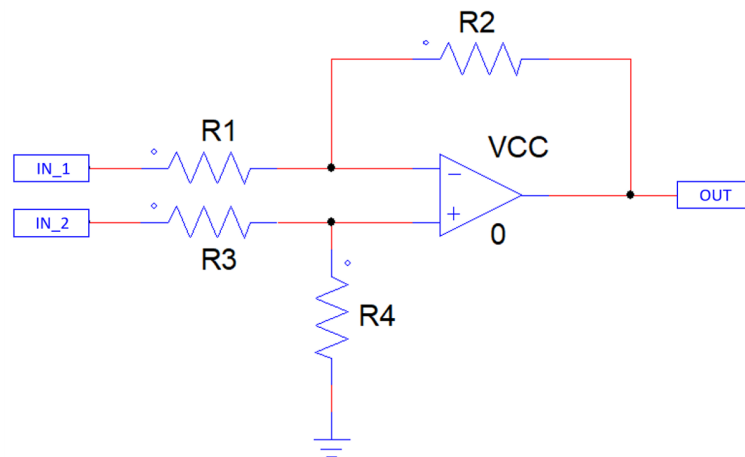
Por esse comportamento de descarga assimétrica, para expandir a vida útil dessas baterias, é recomendado o uso de carregadores do tipo balanceador, capazes de gerenciar a carga de mais de uma bateria ao mesmo tempo e garantir que cada uma seja carregada de maneira proporcional. Em seu estado crítico de operação, a bateria

deve ser removida do sistema e conectada a esses dispositivos externos, responsáveis também por avisar quando os elementos de armazenando de carga estão novamente prontos para utilização. Para o sistema proposto, não está previsto o carregamento *in loco*, obrigando a remoção da bateria e reconexão apenas no momento que estiver preparada para um novo ciclo de descarga.

Em termos físicos, o nível de tensão apresentado por uma bateria pode ser caracterizado como um sinal analógico, que é capaz de assumir qualquer valor ao longo do tempo e, para permitir a leitura dessas grandezas pelo microcontrolador, é necessário fazer uso da interface de conversão analógico-digital do sistema.

Consultando a documentação, é possível perceber que, no caso do *STM32F-103C8TX*, a referência de tensão para a conversão de valores analógicos possui valor padrão de +3,3 V, impedindo a identificação de valores superiores a este nível. Esse limite impede que os níveis descritos para as tensões da bateria sejam diretamente lidos pelo microcontrolador, obrigando a inserção de processos intermediários de ajuste do valor de entrada, para que se encaixe nos limites permitidos. Desta forma, para este projeto foi selecionado um amplificador diferencial como ferramenta de ajuste constante dos níveis de tensão da bateria, com seu modelo ideal de implementação descrito na Figura 21, que segue a relação de ganho estabelecida na Equação 4.1. Por conta da presença de duas células, a quantidade de amplificadores no circuito será duplicada e cada instância terá seus valores de componentes ajustados para o nível de tensão esperado em cada ponto de leitura.

Figura 21 – Circuito Amplificador de Diferença



Fonte: Adaptado de PSIM 9.1.

$$V_{OUT} = \frac{R_2}{R_1}(V_{IN_2} - V_{IN_1}) \quad \forall \quad R_1 \equiv R_3, \quad R_2 \equiv R_4 \quad (4.1)$$

Como apresentado na Equação 4.1, o ajuste da tensão depende diretamente da relação entre os valores de resistores escolhidos e, no caso do primeiro ponto de leitura, que espera por valores de tensão de até 4,2 V, a proporção escolhida para reduzir os valores de entrada foi de 2/3 do valor original, garantindo que para a entrada máxima estimada, o valor convertido seja de aproximadamente 2,8 V. Para encontrar esse fator de redução, foram selecionados valores comerciais de 10 k Ω para os resistores R_2 e R_4 enquanto um valor de 15 k Ω foi designado para o R_1 e o R_3 .

No caso do segundo amplificador, que espera valor máximo de entrada de aproximadamente 8,4 V, uma proporção de 1/3 lhe foi atribuída, com a seleção de valor igual a 10 k Ω para os resistores R_2 e R_4 e 15 k Ω para o R_1 e o R_3 . É interessante notar que o ato de fixar a referência de leitura em 0 V resulta numa simplificação no cálculo da amplificação, já que, aplicando um sinal nulo na entrada IN_1 , nenhum valor de tensão é reduzido do valor da entrada IN_2 , fazendo a saída dependente apenas do valor dessa entrada ajustado pelo coeficiente.

Outro componente que deve ser escolhido é o amplificador operacional utilizado para a topologia. Um elemento bastante comum em aplicações didáticas e disponível em um encapsulamento DIP com 14 pinos, o *LM324* é um amplificador que se adequa para o propósito estabelecido, suportando a tensão necessária e a disponibilidade dos dois amplificadores operacionais necessários para a construção de dois circuitos distintos de leitura.

Com os valores devidamente ajustados, já é possível fazer a conexão aos pinos do *ADC*, que serão vinculados às saídas dos amplificadores diferenciais, com um canal direcionado à aquisição de cada um dos dois valores medidos. Para a leitura do primeiro ponto, será utilizado o canal 5 do *ADC1*, disponível na entrada *PA5*, enquanto a leitura do segundo ponto será realizada pelo canal 4, também do *ADC1*, acessível por meio do pino *PA4*.

Por fim, o último tópico relacionado ao condicionamento da alimentação do sistema diz respeito ao ajuste necessário na tensão fornecida ao barramento principal: como a tensão série equivalente das células conectadas é igual a 7,4 V, é preciso utilizar um mecanismo de conversão e regulação que proporcione em sua saída um sinal de tensão contínua estável de +5 V.

A solução para esse empecilho foi encontrada em módulos comerciais de circuitos que implementam um conversor CC-CC abaixador do tipo *Buck*, com tensão de entrada suportada entre 4,75 V e 23 V e saída ajustável para fornecer entre 1 V e 17 V, além de serem capazes de fornecer uma corrente de até 1,8 A em operação contínua e 3 A para correntes de pico não-repetitivas. Um exemplo de circuito aplicável a este

projeto é o modelo baseado no circuito integrado *MP1584* (MPS, 2011), encontrado em um módulo de fácil utilização e larga comercialização em lojas *online* nacionais, por um custo acessível e formato próximo ao apresentado pela Figura 22.

Figura 22 – Conversor *Step-Down* para ajuste da alimentação do sistema



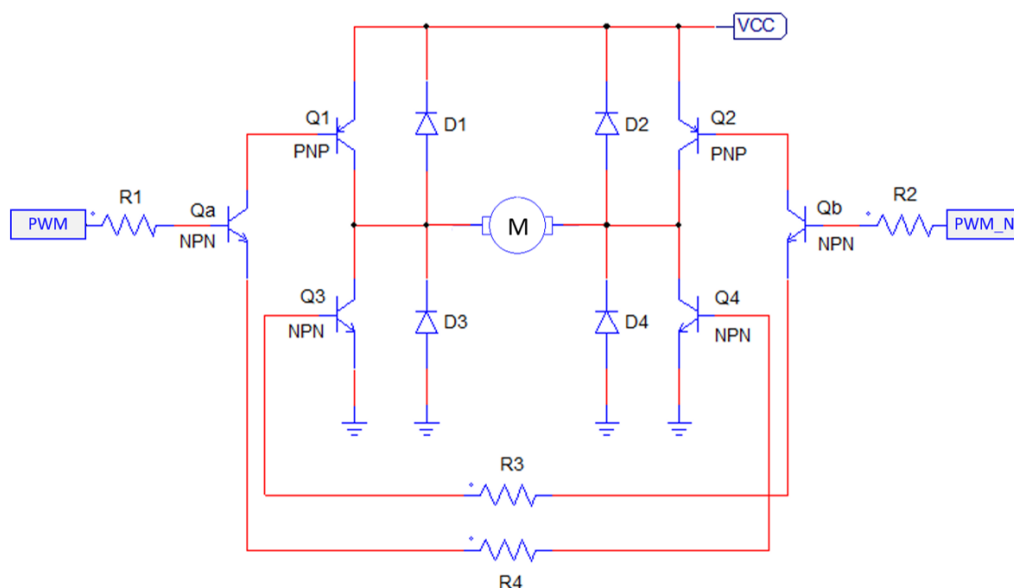
Fonte: <<http://www.eletrogate.com>>.

4.2.1.3 Acionamento dos Motores CC

A seleção dos componentes semicondutores que farão o acionamento dos motores do sistema dependem diretamente das características da carga acrescentada, neste caso, quatro motores de corrente contínua, operando com uma corrente individual estimada em 200 mA. Para a realização destes controles de direção de rotação e velocidade do motor, é possível utilizar uma topologia de circuito como a apresentada na Figura 23, comumente conhecida como ponte-H, controlada por sinais de modulação *PWM* que permitem uma variação na tensão média aplicada sobre os terminais do motor.

Para melhor gerenciar os quatro motores, o acionamento dos motores poderá ser dividido lateralmente na disposição mecânica do carrinho, por meio de duas pontes-H distintas, fazendo com que uma acione os dois motores do lado esquerdo enquanto a outra alimenta os motores do lado direito. Ao determinar que estes circuitos fornecerão potência para duas cargas simultaneamente, estima-se então que a corrente necessária será igual ao dobro da corrente drenada por um único motor. Este parâmetro é importante, pois a corrente que circula no motor deve ser suportada por todas as chaves, já que esta atravessará alternadamente todas as que estiverem acionadas.

Figura 23 – Circuito Ponte-H com Controle PWM



Fonte: Adaptado de PSIM 9.1.1.

Para os transistores Q_1 a Q_4 , que farão o papel de chaves, os modelos utilizados podem ser os transistores de junção bipolar *BD139* e seu complementar *BD140*, transistores *NPN* e *PNP*, respectivamente, capazes de suportar correntes constantes de até 1,5 A e correntes de pico não-repetitivas de até 3 A. Ainda, segundo a folha de dados, suportam tensões reversas de até 80 V. É importante também lembrar que estes elementos passarão por um chaveamento da tensão e que transistores TJB possuem resposta eficiente apenas para chaveamentos até a faixa dos 40 kHz, sendo indicado o uso de outros tipos de chave eletrônica (como *MOSFETs*) para sinais de frequência mais alta. Para um motor CC do tipo utilizado aqui, o acionamento é comumente feito com chaveamento em frequência entre 1 kHz e 10 kHz, dentro do parâmetro suportado por TJBs (ON SEMICONDUCTOR, 2017) (ON SEMICONDUCTOR, 2000).

Os transistores auxiliares Q_a e Q_b não necessitam de capacidades de potência elevadas, já que devem apenas suportar a corrente que circulará nas junções base-emissor das chaves principais que, segundo suas folhas de dados, dependendo do modelo, são capazes de reduzir em até 250 vezes a corrente para o acionamento das chaves, fazendo que, desse modo, seja suficiente o uso de um transistor comum como elemento de acionamento dos demais, como os do modelo *BC548*, capaz de suportar correntes de até 100 mA.

Responsáveis por evitar danos sobre os transistores, a seleção dos diodos deve levar em conta principalmente a frequência de chaveamento utilizada, para que estes consigam responder em tempo hábil às variações de correntes e tensões que são exercidas sobre os componentes que devem proteger. Pensando nisso, foi selecionado

um diodo retificador comum do modelo *1N4007*, capaz de responder adequadamente a frequências de até 1 kHz, comumente utilizada em motores CC de pequeno porte. Se o desejo for de aumentar a frequência de chaveamento a níveis maiores, a recomendação é de utilização de um diodo retificador do tipo *Ultrafast*, como o *UF4007*.

A seleção dos resistores que controlam as correntes de polarização também pode ser realizada de maneira simplificada: considerando que a corrente para o acionamento de dois motores simultaneamente possa chegar a picos de 600 mA em uma condição normal de operação, sob a influencia do pior caso de atuação do TJB com um ganho de 40 vezes para as chaves principais (identificado na folha de dados do *BD139*), a corrente que circulará entre coletor e emissor dos transistores Q_a e Q_b deve ser por volta de 15 mA.

Considerando uma tensão de 3,3 V saída do pino do microcontrolador, e uma queda de tensão de 2,1 V causada pelas junções base-emissor dos três transistores TJB conectados, resta uma tensão de 1,2 V aplicada sobre o resistor de polarização da base do *BC548*. Como a corrente drenada da saída do sinal não deve passar de alguns mA, para não danificar o microcontrolador, limita-se em 1 mA a corrente de base, refletindo no cálculo de um valor de resistência próximo a 1,2 k Ω , mantendo a integridade do sistema. Considerando essa corrente e um ganho mínimo de 110 vezes (verificado na folha de dados do *BC548*), a corrente máxima que seria liberada ao transistor auxiliar é de, no mínimo, 110 mA, suficiente para as especificações estabelecidas anteriormente, que percorrerão este elemento quando o mesmo for acionado.

Por fim, ainda é preciso conectar os sinais de *PWM*, responsáveis por controlar o acionamento das chaves. Analisando as características de funcionamento do *STM32F103C8TX*, é possível identificar que o *timer TIM1* permite gerar automaticamente sinais inversos denominados *TIM1-CH1_N* e *TIM1-CH2_N*, complementares aos sinais gerados pelos canais 1 e 2 de *PWM*, respectivamente, simplificando a programação de controle dos motores. Desse modo, o acionamento da ponte de um dos lados foi conectada ao canal 1 (pinos *PA7* e *PA8*), enquanto o outro lado é comandado pelo canal 2 (pinos *PB0* e *PA9*).

4.2.1.4 Módulos de sensores

Outros elementos bastante importantes no desenvolvimento de estudos relacionados à robótica são os sensores, que permitem ao mundo digital acompanhar as variações do ambiente em que ele se encontra. Em termos de *hardware*, o funcionamento de cada sensor possui sua particularidade e precisa de certos ajustes para garantir a estabilidade na interação com o núcleo de gerenciamento. Para isso, algumas decisões foram tomadas a respeito das diferentes interfaces que compõem o sistema, além da seleção adequada dos pinos para conectar cada uma ao microcontrolador.

Dentre os módulos selecionados para fazer parte do rol de sensores disponíveis para o acompanhamento de informações, dois deles utilizam uma interface de comunicação bastante robusta: o I^2C . O primeiro deles é o sensor de temperatura e pressão, o *BMP180*, que possui em seu corpo apenas quatro pinos, dois para o fornecimento de sua alimentação (*VCC* e *GND*) e os outros dois pertencentes à interface I^2C (BOSCH SENSORTEC, 2013). Para o segundo sensor o cenário é um pouco diferente, disponibilizando um total de oito pinos para a conexão com o mundo externo, conforme evidenciado na Figura 16. Enquanto quatro dos pinos possuem as mesmas funções de alimentação e conexão ao barramento (como o *BMP180*), a presença de outros quatro pinos permite a esse módulo algumas funcionalidades extras (INVENSENSE, 2012).

O pino denominado *INT* é responsável por gerar um sinal lógico que é ativado sempre que um novo dado estiver disponível para leitura, podendo ser utilizado em conjunto com o microcontrolador como estímulo de ativação para uma interrupção em *hardware*. Por ser uma saída, não deverá ser conectado a nenhum nível lógico. Já a entrada marcada como *AD0*, permite a mudança do endereço de barramento ocupado pelo módulo, incrementado em uma unidade quando alimentado em nível lógico alto, permitindo que dois módulos idênticos ocupem o barramento com endereços distintos. Para esta implementação, este pino será conectado ao *GND* (0 V), mantendo seu endereço padrão (INVENSENSE, 2012).

Outro detalhe que precisou ser ajustado para condicionar o funcionamento desta interface diz respeito à alimentação dos módulos, já que, enquanto o *BMP180* possui um limite definido entre +1,8 V e +3,6 V, o módulo *GY-521* permite a entrada de tensões entre +3,3 V e +5 V. Para evitar problemas relacionados a mau-funcionamento e danos ao componentes por diferenças de tensão indesejadas, a opção final foi de alimentar ambos os módulos com uma tensão de +3,3 V, proveniente do regulador de tensão interno ao microcontrolador. Além dos ajustes de tensão, é necessária a adição de resistores de *pull-up* em ambos os fios do barramento, garantindo sua plena operação. O valor dos resistores pode ser selecionado arbitrariamente e, como sugestão, um valor de aproximadamente 1 k Ω se mostra eficiente.

Por fim, ainda é preciso selecionar os pinos que serão utilizados para a conexão do barramento com o microcontrolador. Analisando o diagrama da Figura 32, é perceptível que o *STM32F103C8TX* conta com duas interfaces distintas para conexões de um barramento I^2C e que a interface *I2C1* conta com duas posições diferentes de disposição de seus pinos. Por ter sido escolhido em primeiro lugar, este barramento foi atribuído ao seu primeiro índice, com a conexão realizada por meio dos pinos *PB6* e *PB7* (*SCL* e *SDA*, respectivamente).

Seguindo o processo de condicionamento dos módulos às conexões com o microcontrolador e se mantendo ainda no ramo das interfaces de comunicação, é

necessário configurar os pinos que utilizarão o protocolo *UART* para conexão. Com alterações mais simples que as necessárias para o barramento *I²C*, no caso da *UART*, basta selecionar uma das interfaces disponíveis e conectar os pinos adequadamente.

O sistema que utilizará o protocolo *UART* é o módulo de comunicação por *Bluetooth*, o *HC-05*, e o primeiro passo é fornecer uma condição de alimentação adequada a ele. Em sua folha de dados é possível verificar que o dispositivo é compatível com níveis de tensão de entrada entre +3,6 V e +6 V, tornando adequado o uso da alimentação padrão definida para o *kit* de +5 V, evitando também o uso do nível +3,3 V fornecido pelo microcontrolador, que possui baixa capacidade de fornecimento de potência e já está sendo utilizado por outros dois módulos (ITEAD STUDIO, 2010).

Em sua configuração assíncrona, a *USART* utiliza uma de suas linhas para o envio e outra para a recepção de mensagens, tanto de um lado quanto do outro da conexão. Dessa forma, a ligação entre os dois dispositivos comunicantes ocorre de maneira que a recepção de um esteja preparada para receber as informações emitidas pelo pino de transmissão do outro, fazendo com que as ligações na prática pareçam invertidas. Em resumo, o pino *TX* do módulo deve estar ligado ao *RX* do microcontrolador, enquanto seu *RX* será vinculado ao *TX* da placa de controle.

Além de definir a relação que deve existir entre os pinos, ainda é preciso escolher qual das interfaces do microcontrolador, próprias ao protocolo *UART*, será utilizada. Uma das limitações impostas pelo módulo que fará uso desse meio de comunicação é a tolerância de tensão apresentada pelos pinos usados na troca de informações: consultando a folha de dados do módulo *Bluetooth HC-05* (ITEAD STUDIO, 2010) é possível identificar que, mesmo que a alimentação permita valores maiores de entrada, os pinos *RX* e *TX* suportam apenas uma tensão máxima de +3,3 V e o uso de níveis superiores a este pode causar danos ao dispositivo.

Uma opção viável para resolver o problema consistiria na adição de um divisor de tensão na saída *TX* do microcontrolador, de modo a fornecer ao pino *RX* do módulo um nível ajustado. Entretanto, voltando à Figura 19, é possível perceber que cada um dos pinos de entrada possui uma tolerância máxima e que o diferencial da interface *USART2*, em relação às demais, é uma tolerância menor de seus pinos de conexão, mais especificamente preparada para lidar com sinais de amplitude máxima igual a +3,3 V. Com essas informações em mente, torna-se interessante a seleção dessa interface como forma de cumprir os requisitos de tensão, diminuindo o uso de resistores extras ao conectar o pino *RX* do módulo ao *PA2* e o pino *TX* ao *PA3*.

Voltando à Figura 17, que mostra a aparência física do módulo de conexão *Bluetooth*, é possível verificar que este conta com dois outros pinos além dos utilizados para alimentação (*VCC* e *GND*) e para a comunicação (*RX* e *TX*), denominados *EN* e *STATE*.

O módulo *HC-05* possui dois modos diferentes de operação, um de dados, que permite a troca de informações com qualquer dispositivo *Bluetooth* conectado a ele, e um de configuração, que permite a alteração de parâmetros internos de funcionamento do dispositivo. Para permitir o acesso ao modo de configuração, é utilizado o pino *EN*, que ao receber um sinal de nível lógico baixo ativa essa condição de uso. Se nenhuma tensão for fornecida ao pino, um controle interno automaticamente seleciona o modo de dados como padrão. Como este estudo fará uso das configurações-padrão do módulo, o pino ficará desconectado.

Já no caso do pino *STATE*, este representa uma saída simples, que pode assumir dois estados distintos e anuncia se o módulo se encontra pareado e conectado a algum dispositivo *Bluetooth* (emitindo valor lógico alto). Desse modo, o pino pode ser integrado ao microcontrolador como um estímulo de interrupção e provocar uma rotina de tratamento na manutenção das conexões de dispositivos externos com o módulo. Neste estudo, o uso do pino não se mostrou interessante, portanto, será mantido desconectado.

Outro módulo que deve ser adicionado ao sistema, dessa vez independente de qualquer protocolo de comunicação intermediário, é o módulo de sensor ultrassônico *HC-SR04*. Assim como os outros dispositivos, este também deve ter uma alimentação adequada que, segundo sua folha de dados, deve possuir um nível de tensão de +5 V (MORGAN, 2014). Sobre seus pinos, a definição de posicioná-lo no mapeamento do microcontrolador tem relação direta com suas funcionalidades: além dos dois pinos de alimentação, o módulo conta com um chamado *TRIG*, que funciona como ativador de leitura, e um *ECHO*, que apresentará uma conversão da aquisição em parâmetros elétricos.

Por ser gerenciado pelo controlador, o pino destinado ao estímulo *TRIG* pode ser atribuído a qualquer saída digital de comportamento genérico. Enquanto isso, como o pino de *ECHO* apresenta o resultado na forma de uma largura de pulso, é necessário conectá-lo a uma entrada que seja capaz de interpretar corretamente essa informação. Consultando o comportamento das interfaces do *STM32*, é possível perceber que alguns pinos vinculados aos *timers* do sistema podem ter configurada uma sensibilidade à captura de sinais de entrada, permitindo a correta identificação dessa leitura. O funcionamento do processo que envolve a interpretação desse sinal é feito via processamento em *software* e será descrito em tópicos posteriores. Em resumo, o pino *TRIG* foi atribuído ao *PB10* e o de *ECHO* ao canal 4 do *timer 3*, acessível pelo pino *PB1* da placa *Bluepill*.

Assim como no caso da própria placa do microcontrolador, os módulos de sensores não possuem modelos de símbolo de esquemático ou mesmo *footprints* disponíveis na biblioteca nativa do *KiCad*. Para facilitar o processo deste projeto, foram

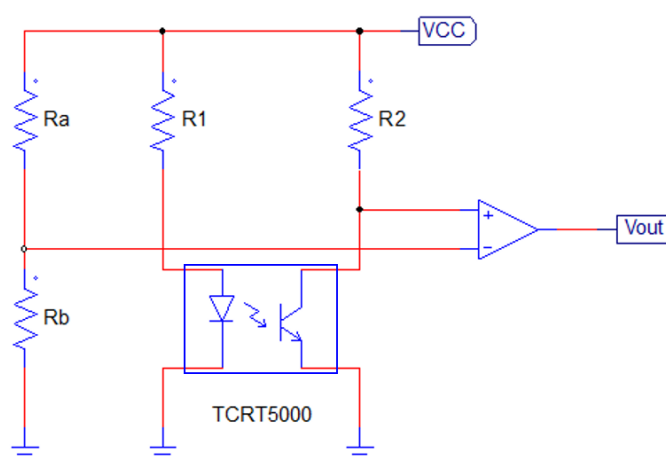
construídos blocos genéricos com descrição dos pinos, um para cada módulo, para utilizar como símbolos de esquemático e, para os *footprints*, foram duplicados os modelos pertencentes a barras de pinos com número adequado a cada um, determinando apenas o espaço ocupado na placa por cada circuito utilizando uma demarcação na camada superior de identificação dos componentes (*Top Silk*).

4.2.1.5 Circuito Seguidor de Linha

O sensor do modelo *TCRT5000*, selecionado para este projeto, é um sensor infravermelho do tipo ativo reflexivo, contando com um emissor e um receptor de ondas infravermelhas, alinhados lateralmente e num mesmo encapsulamento. A recepção neste caso é feita por meio de um fototransistor, revestido por um filtro capaz de bloquear ondas da faixa de luz visível, o fazendo captar apenas frequências da sua própria faixa de emissão. É bastante indicado como sensor para identificação de materiais como papéis e fitas magnéticas, além de funcionar como sensor de posição por detecção de espaços vazios (VISHAY SEMICONDUCTORS, 2019).

Em termos práticos, é possível considerar o funcionamento deste sensor análogo à atuação de mecanismos optoacopladores, com a união de um elemento emissor de ondas eletromagnéticas e de um receptor adequado para realizar um acoplamento e isolamento elétrico de um sinal. Fornecendo uma corrente fixa de polarização para o emissor, este irá emitir sempre uma onda infravermelha de mesma intensidade, com a identificação de uma superfície interpretada por meio da variação de corrente que circula entre o coletor e o emissor do fototransistor, já que a característica de comportamento do sensor é de que, quanto maior a intensidade do feixe captado, maior a corrente estimulada, permitindo perceber variações na reflexão das ondas.

Figura 24 – Circuito de Leitura para o Sensor Reflexivo TCRT5000



Fonte: Adaptado de PSIM 9.1.

Essa variação de corrente pode ser medida por meio de uma tensão aplicada sobre um elemento resistivo, que ao ser percorrido por diferentes valores de corrente, apresenta diferentes valores de tensão e conseqüentemente provoca uma variação na tensão sobre o próprio transistor. Reunindo essas expectativas, foi obtido o circuito da Figura 24.

O valor de tensão obtido pode ser lido de maneira analógica, permitindo compreender de maneira exata a intensidade do feixe infravermelho refletido, entretanto, para o desenvolvimento de um seguidor de linha, entende-se que o leitor deve apenas identificar a presença das linhas de limitação do caminho que devem ser seguidas, somente anunciando se a mesma está presente. O ato de anunciar a presença da linha depende apenas de uma variável digital binária, estabelecendo um limiar de comparação para diferenciar os estados.

Assim, o primeiro parâmetro que pode ser definido para o circuito diz respeito à alimentação fornecida a seus componentes. Considerando que a tensão de +5 V disponível no barramento do sistema é fornecida por uma fonte de potência corretamente dimensionada para suportar todos os circuitos pertencentes ao conjunto, a primeira opção sempre é a de se utilizar esse valor como base para qualquer novo componente adicionado. Desse modo, o valor da tensão no nível *VCC* foi, naturalmente, definido como +5 V. Com a escolha desse valor, passa a ser esperado um nível de tensão de mesma magnitude quando a saída possuir nível lógico alto, causando impacto direto na seleção do comparador necessário para a implementação.

Um modelo de comparador bastante comum em aplicações didáticas é o *LM393* que, segundo sua folha de dados, pode ser alimentado com uma tensão diferencial de até 36 V, permitindo o uso do nível +5 V, escolhido anteriormente (TEXAS INSTRUMENTS, 2018a). Ainda, em seu encapsulamento *DIP* de 8 pinos, apresenta internamente duas unidades de comparação distintas, perfeito para a utilização dos dois circuitos de leitura, um direcionado ao lado esquerdo e outro para o lado direito do robô, estabelecendo as condições mínimas para o seguidor de linha. Desse modo, este foi o componente usado como base para a construção deste circuito na placa final.

Para concluir a inserção destes sensores no circuito, basta fazer a seleção dos demais componentes, resistores de polarização e referenciais de tensão, de acordo com as condições desejadas de funcionamento. Consultando a folha de dados do *TCRT5000* é possível identificar o emissor como um *LED*, capaz de irradiar ondas infravermelhas, que pode ser alimentado seguramente com correntes de até 60 mA. Entretanto, de acordo com aplicações típicas apresentadas (como as condições de teste realizadas pelo próprio fabricante), é possível utilizar corrente mais baixas para a polarização deste elemento, por volta de 10 mA (VISHAY SEMICONDUCTORS, 2019).

Por meio de um gráfico que relaciona a corrente I_F , que ultrapassa o emissor,

e a queda de tensão V_F resultante sobre ele, é possível visualizar que uma corrente igual a 10 mA é responsável por uma queda de tensão de aproximadamente 1,1 V sobre o diodo. Com essa referência, o cálculo da resistência fica bastante simplificado, seguindo os passos estabelecidos pela Equação 4.2.

$$R_1 = \frac{V_{CC} - V_F}{I_F} = \frac{5V - 1,1V}{10mA} = 390\Omega \quad (4.2)$$

Passando para a polarização do fototransistor, a proporção de transdução entre emissor e receptor, verificada em um gráfico da folha de dados do componente, garante que, no limiar de sua região ativa, o transistor seja percorrido por uma corrente de coletor máxima igual a 500 μ A, reflexo da seleção da corrente I_F estabelecida em 10 mA, e que a queda de tensão V_{CE} atinge o valor de 200 mV neste mesmo ponto. Com essas condições de referência, é possível estimar a resistência necessária ao R_2 , por meio da Equação 4.3.

$$R_2 = \frac{V_{CC} - V_{CE(LDS)}}{I_{C(LDS)}} = \frac{5V - 0,2V}{500\mu A} = 9,6k\Omega \quad (4.3)$$

Tendo encontrado um valor de resistência fora dos padrões comerciais, é importante fazer a substituição deste por um resistor de valor próximo, como o de 10 k Ω , utilizado em outros pontos do sistema e um componente bastante comum em circuitos eletrônicos, aumentando a acessibilidade no processo de busca por insumos.

Levando em conta a necessidade de estabelecer um sistema mais dinâmico e personalizável, definir valores fixos para os resistores R_a e R_b , apresentados na Figura 24, não é o cenário ideal. Como forma de evitar esse comportamento estático, estes resistores foram substituídos por um potenciômetro, com os extremos conectados entre o V_{CC} e o GND e seu pino central, de valor variável, ligado na entrada inversora do comparador, permitindo o ajuste do limite de comparação de acordo com o cenário encontrado no ambiente em que o sistema for inserido.

Por fim, a seleção das entradas do microcontrolador responsáveis por fazer a leitura do estado de saída deste circuito deve levar em conta o nível máximo de tensão atingido quando existir um sinal lógico alto, definido como +5 V. Por apresentar apenas valores de lógica binária, podem ser escolhidos quaisquer pinos digitais de propósito geral que suportem esse valor de tensão. Para cumprir a tarefa, foram selecionados os pinos $PA15$ para o sensor esquerdo e $PB3$ para o sensor da direita.

4.2.1.6 Circuitos Indicadores de Sinalização Visual e Sonora

Criar uma plataforma que tem por finalidade o desenvolvimento de novas soluções, com o intuito de ser indefinidamente personalizada, reflete no surgimento de

uma preocupação com mecanismos que facilitem os processos intermediários inerentes à construção de novas aplicações. Enquanto as ferramentas, a nível de *software*, podem ser incorporadas ao código de maneira simples, adaptações posteriores às configurações de *hardware* são mais complicadas de se executar. Como mencionado anteriormente, em sistemas embarcados, plataformas que comumente não possuem interfaces visuais de interação, é maior a dificuldade na depuração do código desenvolvido, por conta da falta destes meios de visualização, dependendo ainda mais dessas ferramentas auxiliares selecionadas.

Com esse propósito, foram selecionados dois tipos de componentes que podem servir como ferramentas emissoras de *feedbacks*, de fácil identificação e com implementação simplificada, podendo ser configuradas para a notificação da ocorrência de eventos diversos: um conjunto de LEDs para notificações visuais e um *Buzzer* passivo para emitir sinais sonoros.

O conjunto proposto para a utilização dos *LEDs* como interfaces de depuração conta com três elementos controlados por pinos de saída de propósito genérico disponibilizados pelo microcontrolador. Por meio de intervenções em *software*, os pinos determinam valores lógicos de saída que fazem o *LED* designado acender ou apagar, dependendo da configuração atribuída.

Eletricamente falando, um *Light-Emitter Diode (LED)* é um diodo que possui a capacidade de emitir ondas eletromagnéticas do espectro de luz visível quanto corretamente polarizado. Para esses componentes, uma configuração adequada é obtida quando a fonte alimenta o componente em modo de polarização direta, com uma diferença de potencial positiva entre seus terminais chamados Ânodo (A) e Cátodo (K). Em termos ideais, a aplicação direta de um nível de tensão é suficiente para que o circuito funcione corretamente, entretanto, por conta dos limites elétricos não-ideais de um *LED*, existe a necessidade de um controle da corrente que o atravessa, evitando danos por sobrecarga.

Ainda, pensando que no sistema aqui projetado a tensão fornecida ao *LED* será proveniente de um dos pinos de saída do microcontrolador, o controle de corrente também faz-se necessário para evitar que a corrente que polariza o *LED* seja superior ao limite de fornecimento do próprio microcontrolador. A forma mais simples de controle de corrente em um circuito elétrico é a adição de um resistor, que pode ser conectado em série ao diodo, na saída do microcontrolador.

Para estimar um valor adequado de resistência, basta identificar a tensão de saída no pino do microcontrolador e descontar a queda de tensão V_F , que deve ocorrer pela natureza do *LED*, com valores entre de 1,5 V a 1,8 V. Tipicamente, diodos emissores de luz possuem uma tolerância para a corrente I_F , de polarização, que suporta valores máximos entre 20 e 30 mA. Entretanto, ao ser alimentado por uma das

saídas do microcontrolador, capaz de fornecer uma potência baixa da ordem de poucos mA, é importante limitar expressivamente a corrente que percorre o circuito, com o ônus de reduzir a luminosidade emitida por este elemento.

É possível utilizar circuitos auxiliares com chaves eletrônicas para polarizar os LEDs a partir de correntes mais altas, contudo, para este estudo, essa modificação refletiria na inserção de um maior número de componentes ao passo de pouca evolução de comportamento, tornando-se pouco viável. Desse modo, limitando a corrente de polarização em 3 mA, valor seguro para o microcontrolador, e considerando a tensão de saída V_{OUT} de 3,3 V, o cálculo da resistência R_{LED} pode ser feito a partir da Equação 4.4.

$$R_{LED} = \frac{V_{OUT} - V_{F(mx)}}{I_F} = \frac{3,3V - 1,8V}{3mA} = 500\Omega \quad (4.4)$$

Por não ser comercial, o valor calculado para a resistência de polarização dos LEDs deve ser substituído por valores próximos, como 470 Ω , que já é empregado em outras partes do circuito, facilitando o processo de compra dos componentes necessários para a montagem dos circuitos.

Assim como o LED, para que um *buzzer* emita sons e funcione como elemento de *feedback*, é importante a adição de um circuito responsável por polarizar corretamente este elemento e fornecer a energia necessária para sua atuação.

Em essência, um *buzzer* é um transdutor que faz uso do chamado efeito piezoelétrico reverso para a geração de sons. Materiais piezoelétricos são conhecidos por terem a curiosa característica de gerar uma diferença de potencial em sua extensão quando sofrem algum tipo de deformação mecânica. Outro fato curioso é que o comportamento contrário também é válido e o efeito piezoelétrico reverso diz que um material deste tipo sofrerá deformações mecânicas se uma diferença de potencial lhe for aplicada.

Segundo os princípios da física para ondas mecânicas, uma deformação repetitiva de uma superfície e sua interação com o ar que a circunda pode gerar vibrações que, em uma faixa de frequência audível, gera sons característicos identificados pelo ouvido humano. Desse modo, o funcionamento de um *buzzer* é facilmente compreendido ao perceber que a variação constante da tensão aplicada sobre o piezoelétrico gera vibrações proporcionais e, se o controle dessas vibrações garante uma frequência mecânica na faixa audível, o dispositivo emitirá sons.

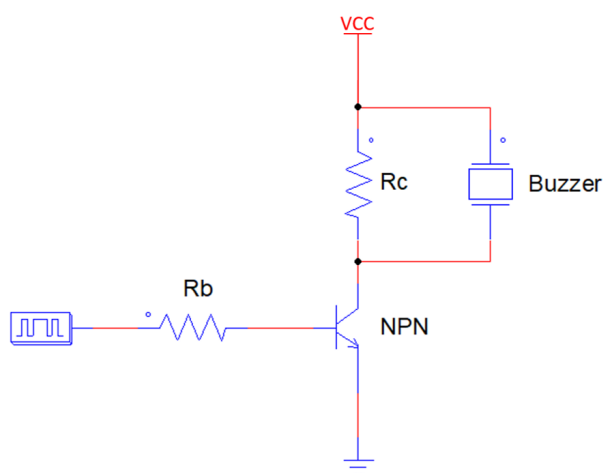
Comercialmente, um *buzzer* pode ser de dois tipos: ativo ou passivo. Embora o encapsulamento dos dois tipos seja semelhante, a diferença entre esses componentes é definida basicamente pela complexidade do circuito interno que o acompanha, já que em um *buzzer* ativo há a presença de um circuito oscilador de frequência fixa, pronto para causar vibração audível no piezoelétrico assim que for alimentado,

enquanto para um *buzzer* passivo nenhum circuito interno é responsável por essa tarefa, necessitando de uma configuração auxiliar externa para realizar o trabalho. As vantagens ou desvantagens ficam à cargo da aplicação, pois *buzzers* ativos são recomendados para sons não-melódicos, como alarmes e sinalizadores (já que a frequência do oscilador interno não pode ser modificada), enquanto *buzzers* passivos podem ser conectados a circuitos capazes de temporizar as vibrações e até mesmo criar melodias simplificadas.

Para este projeto, de modo a permitir uma personalização maior do som emitido pelo *buzzer*, foi escolhido um componente do tipo passivo como base para os avisos sonoros, construindo um circuito responsável por controlar as vibrações sobre o piezoelétrico. Para estimular essas vibrações, a forma mais comum de geração de frequências em um microcontrolador é por meio de um sinal *PWM*, como visto anteriormente para o controle dos motores. Entretanto, diferentemente do *LED*, a corrente de polarização necessária para alimentar um *buzzer* tende a apresentar valores maiores, muitas vezes exigindo uma potência excessiva das saídas de *PWM* do microcontrolador, impedindo uma conexão série diretamente nesta saída.

Como maneira de evitar danos elétricos a ambos os sistemas, a mesma ideia de chaveamento vista nos circuitos de controle dos motores foi selecionada para alimentar o *buzzer*: ao invés de ser conectado em série ao *buzzer*, o sinal de *PWM* passa a servir como acionamento na base de um transistor que atua como chave eletrônica, alimentando o circuito com a tensão do barramento da fonte externa do sistema, com capacidade maior de fornecimento de energia. Essa forma de alimentação também é validada ao consultar a folha de dados de *buzzers* passivos da fabricante *TDK Electronics*®, que sugere o uso de um transistor para a polarização do *buzzer* conforme o esquema apresentado na Figura 25 (TDK CORPORATION, 2011).

Figura 25 – Circuito recomendado para o controle de um *buzzer* passivo



Fonte: Adaptado de PSIM 9.1.1.

No circuito, o resistor R_c é responsável pela carga e descarga do elemento piezoelétrico e, segundo a folha de dados mencionada, um valor de resistência igual a $1\text{ k}\Omega$ é suficiente para garantir a eficiência do conjunto. Enquanto isso, a tarefa desempenhada pelo resistor R_b é de controle da corrente de saída do pino do microcontrolador, o mesmo que alimenta a base do transistor. Para dimensionar o resistor R_b , é preciso saber que um valor comum para a corrente de alimentação de um *buzzer* é estimada entre 30 mA e 50 mA e também que a corrente aplicada à base do transistor limita a corrente máxima que pode circular entre o coletor e o emissor deste, dependendo de um ganho intrínseco ao elemento semiconductor, assumido como 100 vezes para cálculos mais imprecisos ($I_c = \text{Ganho} * I_b$).

Se for definido que a corrente mínima a percorrer o sistema para um funcionamento adequado do *buzzer* é de 50 mA , então o nível mínimo na base do transistor deve ser de pelo menos $500\text{ }\mu\text{A}$. Com a saída do microcontrolador assumindo tensão de $3,3\text{ V}$ em nível lógico alto, ao descontar os $0,7\text{ V}$ da queda na relação base-emissor, tem-se que a tensão aplicada ao R_b estará próxima a $2,5\text{ V}$. Com esses valores de tensão e corrente definidos, estima-se que o valor máximo de resistência para este ponto de circuito deve ser de aproximadamente $5\text{ k}\Omega$. Para uma margem de corrente que permita até 100 mA circulando, foi escolhido para este projeto uma resistência para R_b igual a $2,2\text{ k}\Omega$.

4.2.1.7 Circuitos auxiliares

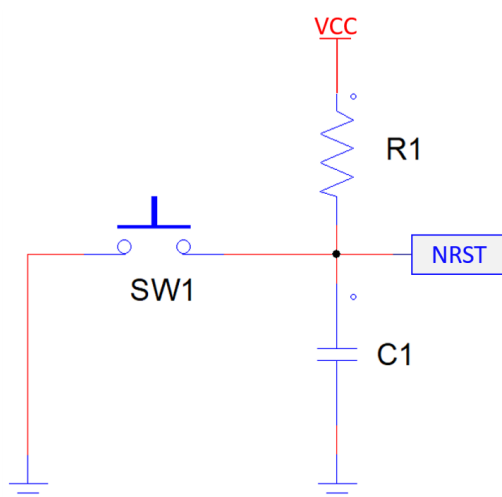
Assim como os elementos de *feedback* apresentados como facilitadores do processo de desenvolvimento, alguns componentes auxiliares podem ser adicionados para facilitar a identificação de problemas de implementação ou mesmo rapidamente permitirem reorganizar o sistema em uma configuração diferente, abrindo espaço para o uso de diferentes possibilidades em um mesmo cenário.

Neste sentido foram adicionadas duas barras de pinos paralelas aos conectores da base do microcontrolador, permitindo mais uma forma de acesso direta a essas conexões e incentivando a posterior personalização de entradas e saídas ainda não utilizadas. Também, a adição dessas interfaces foi pensada para facilitar acesso para medições, feitas durante o processo de configuração dos sistemas selecionados.

Em casos mais extremos, quando o sistema apresenta comportamento indesejado ou necessita que sua operação seja reiniciada, é comum a existência de interfaces de interação do usuário para interferir na rotina corrente, interrompendo imediatamente sua execução. Na placa *Bluepill*, um dos pinos disponíveis está conectado a uma entrada com este propósito específico no microcontrolador: a conexão marcada como *NRST* permite reiniciar imediatamente todo o sistema de controle ao ser estimulada com um sinal de nível lógico baixo. Para garantir a estabilidade destes sinais

atribuídos, um circuito auxiliar pode ser acoplado ao pino, visualizado na Figura 26 e composto por um botão tátil em paralelo a um capacitor, o que estabelecerá um nível lógico alto no momento de alimentação do sistema (derivado do capacitor que se carregará) e um nível lógico baixo no momento que o botão for pressionado, resultando no *reset* desejado.

Figura 26 – Circuito auxiliar para o *reset* do sistema



Fonte: Adaptado de PSIM 9.1.1.

Ao observar a placa *Bluepill*, é possível perceber a presença de um pequeno botão, destinado ao mesmo propósito, mas, por conta de seu tamanho diminuto, muitas vezes de difícil acesso ao usuário. Este circuito é opcional e sua remoção não impacta no funcionamento do restante do sistema.

A finalização de todo esse desenvolvimento e conexão dos circuitos deve resultar em um diagrama de esquemático que representa a abstração do sistema como um conjunto de circuitos teóricos, importante por servir como base para a posterior construção do *layout* de placa, mas também por apresentar-se como uma referência no processo de montagem, já que evidencia as relações entre os códigos de referência de cada componente e seu valor prático, facilitando a montagem artesanal do *hardware*. Na Figura 27, é possível identificar o esquemático exportado da ferramenta *Kicad*, contendo os subconjuntos discutidos nos tópicos apresentados anteriormente.

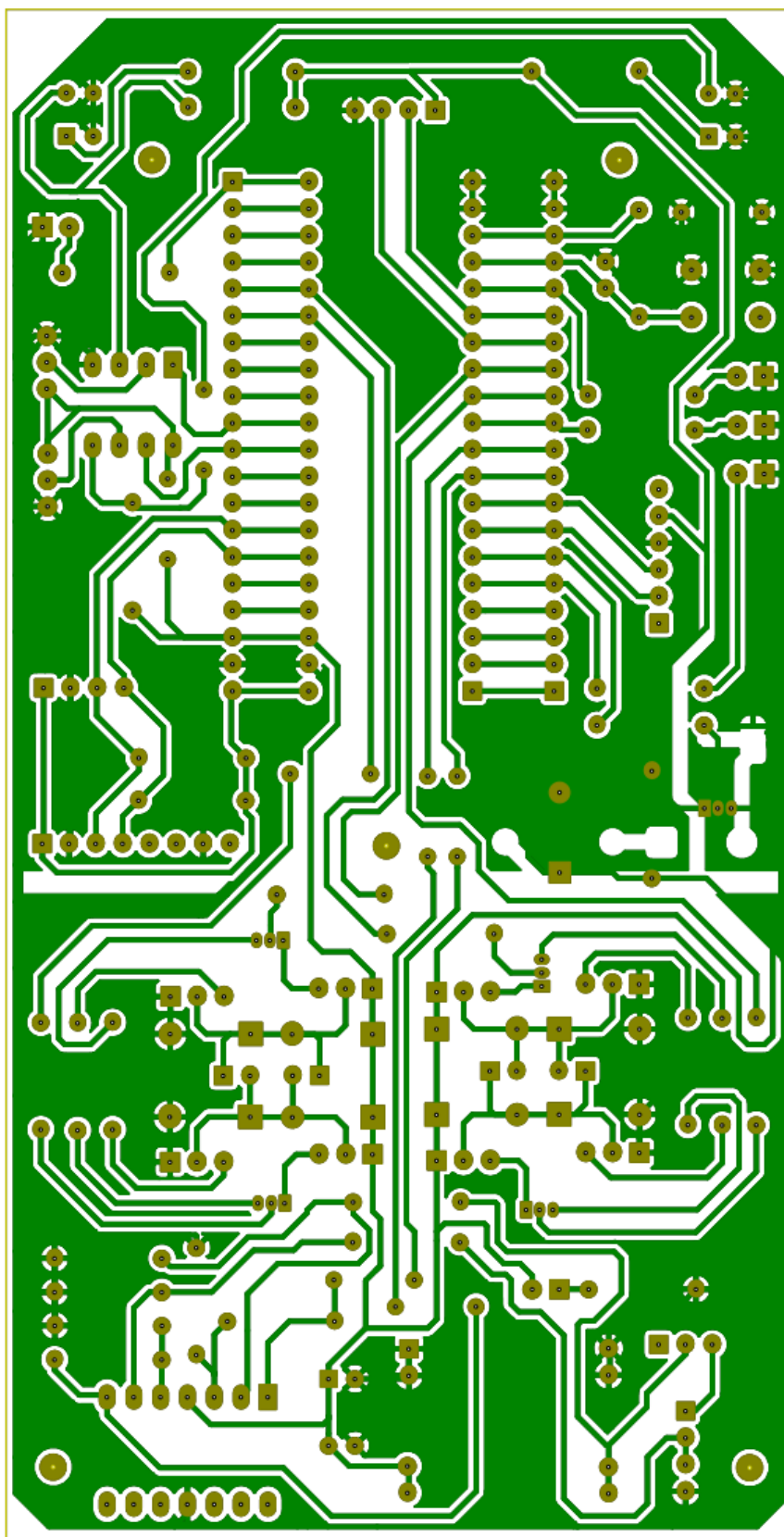
Além do diagrama de esquemático, outro documento que é de grande importância para o projeto de uma placa de circuito impresso é a chamada Lista de Materiais, do inglês *Bill of Materials (BOM)*, responsável por identificar todos os componentes presentes em uma placa. Esse documento é bastante comum, principalmente em âmbito industrial, pois facilita a identificação dos insumos necessários, acelerando a etapa de compra destes elementos. O *software* permite nativamente a geração destes documentos, e a lista relacionada a este projeto pode ser vista na Tabela 12.

4.2.2 Desenvolvimento do *Layout* de PCI do Sistema Proposto

Depois de estabelecer a distribuição dos componentes e ligações pelo diagrama de esquemático, o próximo passo é desenhar o *layout* de uma placa de circuito impresso que fará a tradução dessas conexões ideais em trilhas de cobre, permitindo a construção física do circuito final. A execução deste processo consiste em encontrar as posições adequadas para os conjuntos pertencentes ao sistema, respeitando condições de potência, eficiência e dimensões máximas.

O projeto de uma placa de circuito impresso deve seguir boas práticas de construção, verificando as características críticas do sistema, tornando os comportamentos elétricos indesejados menos frequentes e garantindo uma melhor condição de funcionamento para o sistema. Pensando nesses critérios, o *layout* da placa necessário para implementação do circuito aqui descrito foi desenvolvido e o seu resultado pode ser identificado na Figura 28, contendo as trilhas e os *pads* referentes a todos os componentes pertencentes ao conjunto final.

Figura 28 – Layout da PCI para o sistema proposto

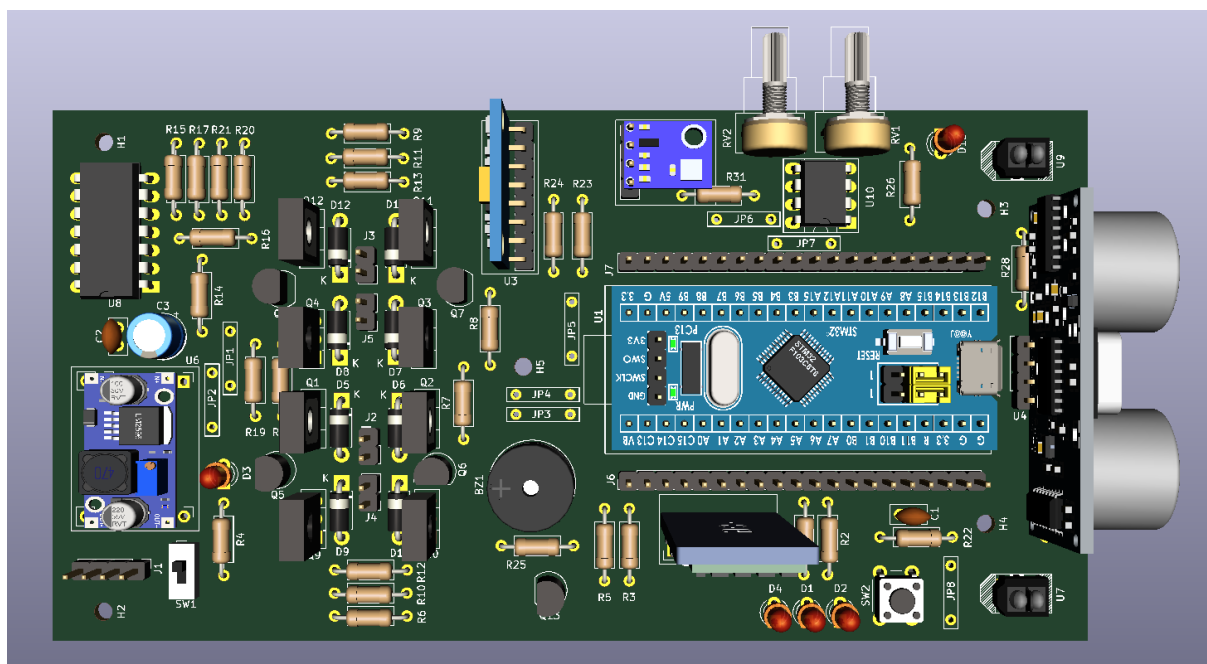


Fonte: Adaptado de *KiCad 5.0.0*.

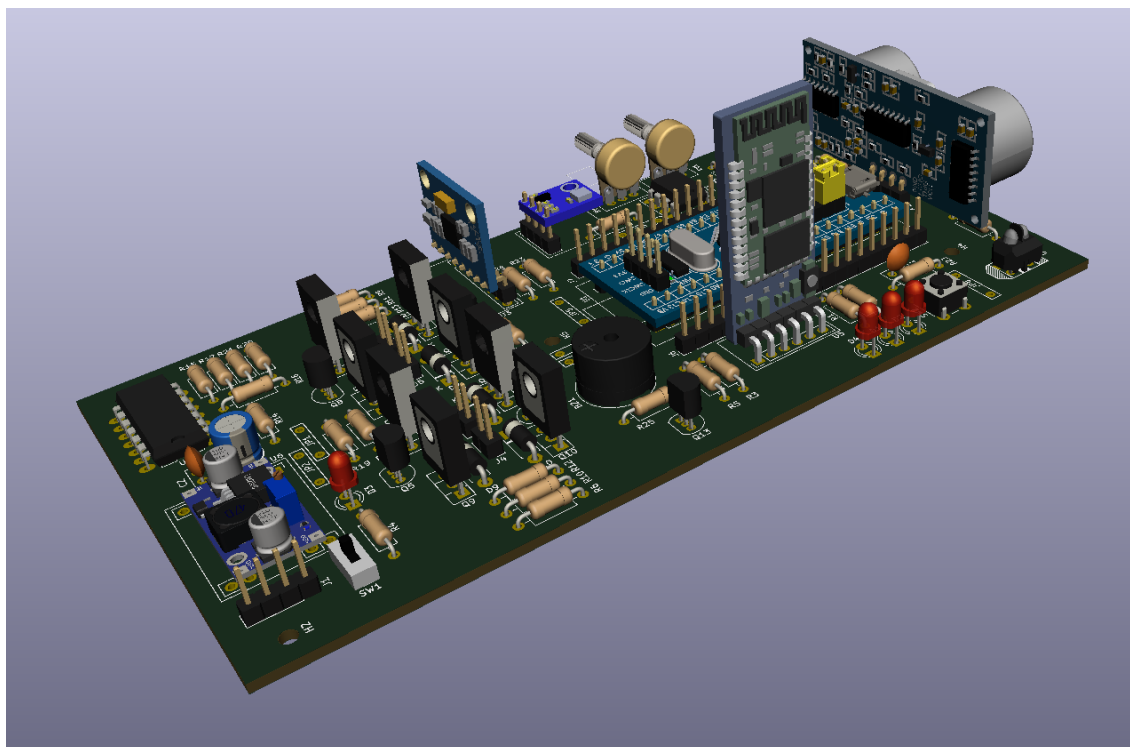
É comum em ferramentas de desenho voltadas a circuitos elétricos a apresentação de uma visualização 3D do *layout* criado, permitindo verificar algumas características de organização dos componentes, que pode ser útil em situações em que a altura máxima é um fator crítico, sendo bons exemplos disso aplicações que envolvem a inserção das placas em gabinetes mecânicos, normalmente apresentando limitação em suas dimensões físicas. A utilização desta funcionalidade também pode servir bem para identificar relações de proximidade entre os componentes, podendo, por exemplo, evitar que efeitos térmicos ou eletromagnéticos causem danos aos elementos próximos.

Neste estudo, o modelo de visualização 3D facilita o processo de imaginar a configuração da placa mesmo antes de se completar o processo de confecção prática, permitindo o entendimento da densidade de componentes e o posicionamento adequado dos sensores. Nas imagens avistadas na Figura 29 e na Figura 30, são apresentados três pontos de vista diferentes do modelo 3D resultante, permitindo imaginar o produto final esperado da montagem deste projeto.

Figura 29 – Vista superior do modelo 3D para a PCI do sistema proposto



Fonte: Adaptado de *KiCad 5.0.0*.

Figura 30 – Vista lateral do modelo 3D para a PCI do sistema proposto

Fonte: Adaptado de KiCad 5.0.0.

4.3 Desenvolvimento de *Firmware* e *Software*

Depois de definir todos os aspectos físicos dos circuitos que compõem o *kit* didático, o próximo passo reside na busca pela aplicação que fará o gerenciamento de todos os subconjuntos agregados a este sistema, realizando as leituras necessárias das variáveis de interesse, atuando como controle para a interação do robô com o ambiente e permitindo a comunicação com dispositivos externos. Nessa fase de desenvolvimento, o foco principal reside em inicializar corretamente cada interface e criar uma rotina para a manutenção de suas atividades recorrentes.

Novamente, por depender diretamente de recursos de programação, o uso de artifícios computacionais é imprescindível para essa etapa do projeto. A escolha das ferramentas adequadas deve ser por meio da avaliação de compatibilidade com a linguagem e a plataforma de desenvolvimento que está sendo utilizada, neste caso, a placa *Bluepill* baseada em microcontroladores de modelo *STM32F103C8TX*, fabricado pela *ST Microelectronics*. Em sua documentação oficial, a *ST* recomenda a utilização de alguns *softwares* proprietários gratuitos para o desenvolvimento de soluções direcionadas a esses modelos de *hardware*, com plataformas intuitivas e completas, tanto para o gerenciamento quando execução do projeto.

Nos tópicos discutidos a seguir, uma série de apontamentos envolvendo a

utilização destes *softwares* de desenvolvimento, técnicas de implementação e especificidades do microcontrolador serão discutidas, tentando trazer mais clareza à sequência de escolhas que culminaram na estrutura final alcançada.

4.3.1 Inicialização do Sistema

A configuração inicial de um microcontrolador é bastante importante e, por vezes, pode ser um pouco confusa se feita manualmente, pois depende da interação com uma série de registradores internos e da compreensão das informações presentes no manual de operação da plataforma. Para facilitar esse processo e todo o gerenciamento das funcionalidades desejadas para o microcontrolador, é comum a existência de plataformas desenvolvidas pelo próprio fabricante das unidades de controle, capazes de facilitar a personalização para o funcionamento do sistema, tornando-a mais rápida e clara.

Com a *ST Microelectronics* esse cenário não é diferente: o *software* proprietário *STM32Cube* que, segundo o próprio distribuidor, funciona como um ecossistema completo para todos os microcontroladores e microprocessadores da família *STM32*, com soluções em *software* tanto para usuários que precisam ferramentas para todas as etapas do desenvolvimento ou aqueles que já utilizam *IDEs* alternativas para a criação de soluções.

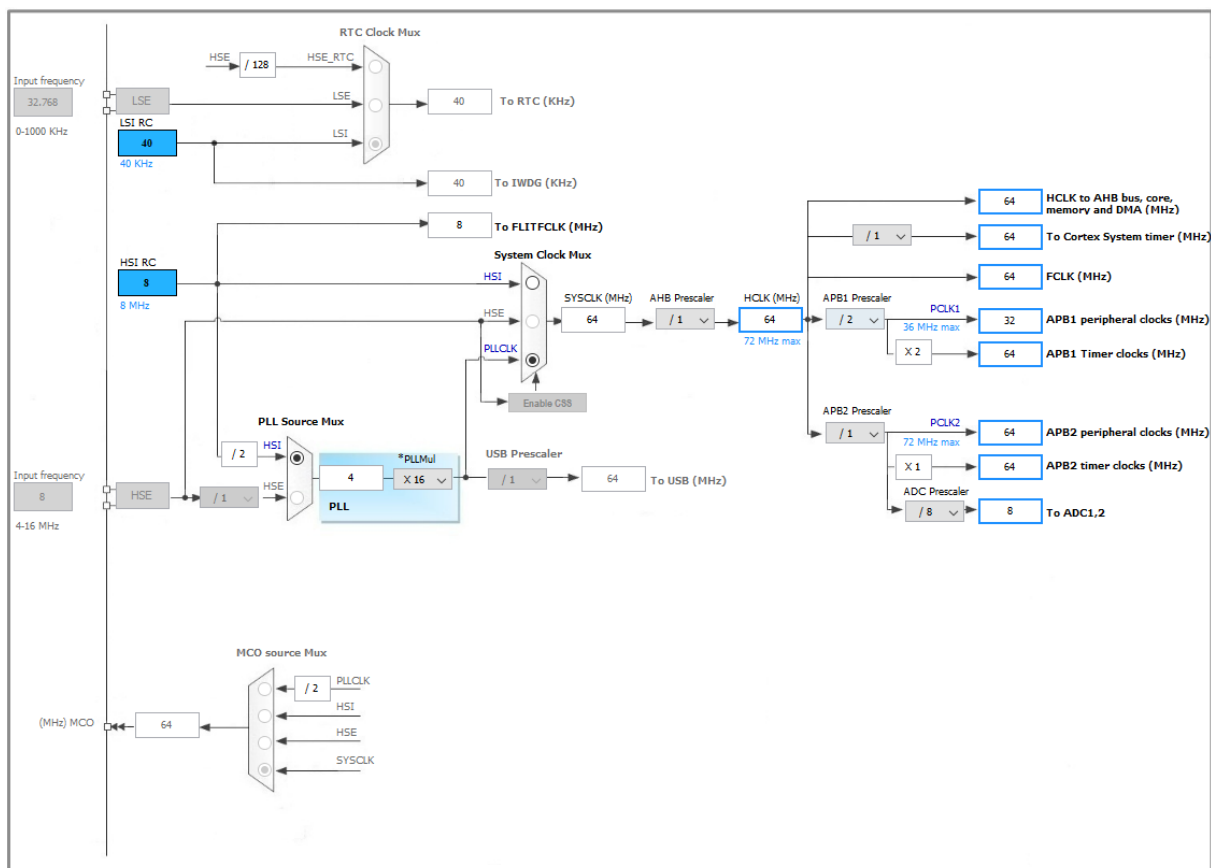
Para este projeto, uma das ferramentas que compõem este ambiente de soluções, mais especificamente a plataforma *STM32CubeMX*, será de grande auxílio na configuração do contexto, permitindo o uso de uma interface gráfica para seleção dos parâmetros de sistema desejados. Neste processo, por exemplo, podem ser definidos os valores e os modos de operação vinculados a todos os periféricos disponíveis, tudo de maneira visual e intuitiva. A versão da solução utilizada aqui neste desenvolvimento foi a v4.26.1.

Sendo assim, o primeiro passo é criar um novo projeto-base que seja compatível com o *STM32F103*, tarefa facilitada pelo *STM32CubeMX*, que apresenta uma aba com todos os microcontroladores disponíveis e outra com todos os *kits* de desenvolvimento comercializados, facilitando a identificação da plataforma. Essa seleção implica no acesso a bibliotecas dedicadas a cada modelo de microcontrolador, habilitando uma camada de abstração para as interações com o *hardware* e evitando que seja obrigatória a configuração do dispositivo por meio dos registradores individuais. No menu inicial, basta selecionar um *New Project*, filtrar pelo dispositivo especificado, no caso o *STM32F103C8Tx*, e iniciar a configuração.

Ao clicar em *Start Project*, o usuário é imediatamente direcionado a uma janela com quatro abas distintas: *Pinout*, *Clock Configuration*, *Configuration* e *Power Consumption Calculator*.

Atuando como peça chave no desempenho da aplicação, o *clock* do sistema é o sinal de sincronia que estabelece a temporalidade nas operações realizadas e, por meio da aba *Clock Configuration*, vista na Figura 31, é possível selecionar o valor desejado de frequência deste estímulo.

Figura 31 – Ajuste do sinal de *clock* para sincronia do sistema



Fonte: Adaptado de STM32CubeMX v4.26.1.

A configuração permitida pelo *software* conta com um auto-ajuste, onde a frequência desejada de operação é inserida no campo *HCLK* e a ferramenta fica responsável por calcular automaticamente os parâmetros que possam levar à frequência mais próxima da selecionada. Caso não consiga encontrar exatamente o valor descrito, o sistema notifica e realiza o ajuste para uma frequência alternativa possível: como um exemplo, na configuração deste projeto, ao tentar inserir o valor máximo de operação suportado (72 MHz), o *STM32CubeMX* notificou que não conseguiria atingir essa configuração com as entradas e parâmetros selecionados, questionando se poderia realizar o auto-ajuste. Após esse balanceamento, a frequência resultante foi de 64 MHz, gerada pelo sistema de *PLL* e derivada da fonte *HSI*.

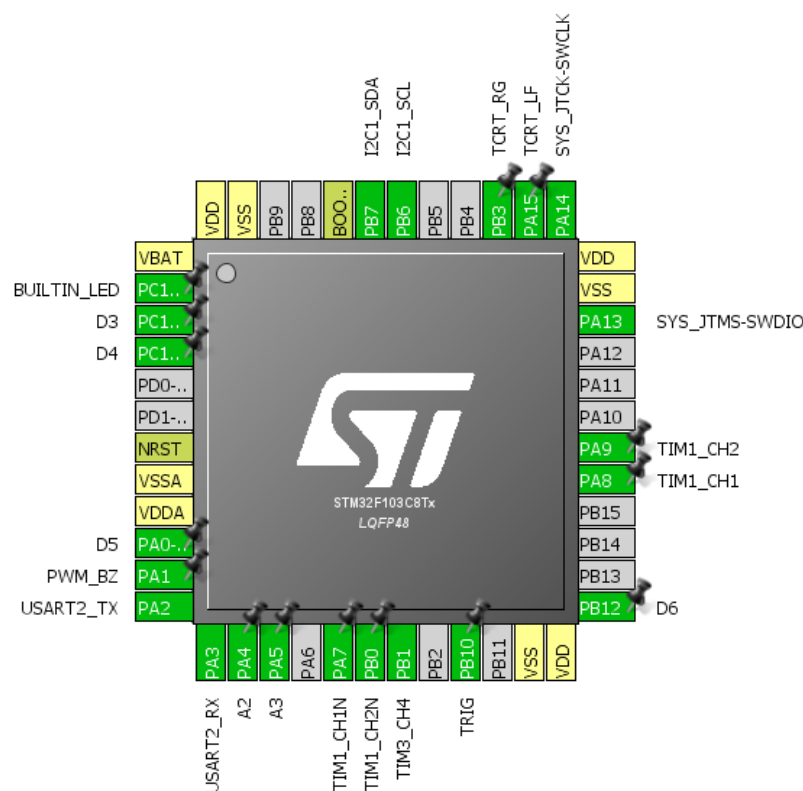
Ainda, verificando o diagrama, é possível perceber que as frequências

associadas aos subsistemas também podem ser ajustadas, por meio de caixas de seleção destinadas a diferentes sinais de *clock* espalhados pelo sistema, como para circuito de gerenciamento do acesso à memória, dos *timers*, dos periféricos e do conversor analógico-digital.

4.3.2 Configuração dos *GPIO*

Outra aba disponível na janela principal do *STM32CubeMX* é nomeada *Pinout* e assim como a configuração do *clock*, possui papel fundamental na atribuição das condições de utilização das funcionalidades do sistema. No diagrama de pinos presente nesta aba estão distribuídas todas as conexões encontradas no circuito integrado que implementa o *STM32F103C8TX* e, ao clicar sobre um dos pinos disponíveis, um menu de seleção aparece, evidenciando quais interfaces fazem uso deste como uma entrada ou uma saída. Como o número de pinos de um microcontrolador é reduzido, as interfaces são obrigadas a compartilhar conexões, tornando-as mutuamente exclusivas se conectadas ao mesmo ponto. Após configurar corretamente todas as entradas e saídas, conforme as definições feitas durante a confecção da placa, a visualização do diagrama assumiu o estado visto na Figura 32.

Figura 32 – Seleção dos pinos de entrada e saída da plataforma *STM32F103C8TX*



Fonte: Adaptado de STM32CubeMX v4.26.1.

Os pinos *PA13* e *PA14*, embora não mencionados anteriormente, podem ser atribuídos a um propósito específico e não fazem parte da sequência pertencente aos barramentos paralelos de 20 pinos nas laterais da placa *Bluepill*. Tomando como referência a Figura 19, é possível identificar que essas entradas estão conectadas à sequência de pinos dispostos na parte inferior da placa, que é utilizada basicamente para a gravação do programa à memória interna e como interface serial para conexão a ferramentas externas de depuração de código. Desse modo, é fácil entender porque a ativação desses pinos é feita pela seleção do modo *Serial Wire* no campo de *Debug* para a interface *SYS*, encontrada no menu lateral.

Outro pino não mencionado anteriormente e selecionado para esta aplicação é o *PC13*. A configuração deste pino acontece pelo fato de estar conectado diretamente ao *LED* da placa *Bluepill*, permitindo o controle do acendimento deste e sua utilização como um elemento extra de *feedback* para a aplicação. Para habilitar o pino como controle do *LED* basta selecionar a função *GPIO_Output* no menu de seleção. A atribuição de nomes customizados também é possível, podendo ser feita no menu *Configuration* ao selecionar a configuração do *GPIO* e alterar os nomes na coluna *User Label*. Outra maneira rápida de renomear é por meio do diagrama mostrado na Figura 32, clicando com o botão direito do *mouse* sobre o pino e selecionando *Enter User Label*.

4.3.3 Temporizações

Para suprir algumas das necessidades de temporização da aplicação, foi desenvolvida uma implementação em *software* de temporizadores, ideal para controlar o tempo de execução de tarefas com baixo teor de criticidade, como leitura de sensores para variáveis de menor taxa de variação ou envio de eventos de atualização de *status* do sistema. Pensando nesses critérios de implementação é possível definir uma estrutura simplificada com variáveis de interesse para um temporizador, como visto no Código A.1.

Na estrutura definida em linguagem *C*, três variáveis distintas descrevem completamente um temporizador: uma variável de referência posicional *start_tick*, que armazena o instante em que a contagem se iniciou; um limite de ciclos *timeout*, que é utilizado como comparação para identificar o estouro de uma contagem; o modo de operação *timer_mode*, responsável por definir se o timer é periódico ou se só é configurado para estourar uma única vez. A interação entre essas variáveis permite estabelecer intervalos de ocorrência de eventos de maneira bastante simplificada e, por conter apenas variáveis simples, permite instanciar um número indefinido de temporizadores simultâneos.

As funções que permitem executar todo o controle sobre os temporizadores

também possuem características bastante simplificadas, apresentando rotinas responsáveis por inicializar a estrutura, começar uma contagem, parar uma contagem e identificar um estouro. Com essas quatro sequências de execução já é possível estabelecer temporizações funcionais para o sistema, sendo uma possibilidade de implementação delas apresentada no Código A.2.

Para iniciar uma contagem, basta atribuir à variável *start_tick* o valor numérico que representa o instante que será utilizado como referência de início, podendo ser um valor de contagem de ciclos proveniente de algum dos periféricos dedicados de temporização ou mesmo da contagem global de ciclos do sistema, acessível pela função *HAL_Get_Tick()*. Ao iniciar a contagem, é também preciso fornecer um *timeout*, que define o intervalo antes do estouro e é uma grandeza posicional, verificada pela quantidade de ciclos que se passaram desde o instante marcado em *start_tick*.

Depois de iniciado o *timer*, a verificação progressiva da passagem do tempo permite identificar se limite de contagem já foi atingido ou não. Evidenciada na função *Timer_Timeout*, a verificação deste estouro depende de uma comparação booleana com dois termos, o primeiro ditando que o valor de início é diferente de zero, o que significa que o *timer* está em operação, e o segundo termo derivado da comparação entre a diferença de ciclos calculada entre o instante atual, fornecido como parâmetro, e o instante inicial, verificando se esta é maior que o *timeout* definido na inicialização. Se ambas as condições forem verdadeiras, um estouro do timer é confirmado e o *timer* é gerenciado, reiniciando apenas se o modo *CONTINUOUS* estiver habilitado. Do ponto de vista da aplicação, o estouro é sinalizado por meio do retorno da função, que é o resultado booleano da comparação descrita anteriormente.

Ainda no Código A.2, uma função adicional é apresentada, *timeDiff()*, responsável por calcular o intervalo de tempo decorrido desde o início da contagem até o instante atual, verificação importante para a comparação booleana na identificação do estouro de contagem. De maneira prática, a implementação desse cálculo foi feita com base em variáveis genéricas, permitindo identificar a diferença temporal de dois instantes quaisquer recebidos como parâmetro, podendo ser utilizada para outras aplicações ao longo do projeto, conforme necessário.

Para facilitar o entendimento sobre o uso dessas estruturas, o Código A.3 mostra um exemplo simplificado de utilização de um temporizador implementado em *software*. Neste exemplo, um *timer* de depuração é inicializado em modo de operação contínua e sua passagem de tempo é baseada no contador de ciclos global. Como cada ciclo do contador global equivale a um instante temporal de 1 ms, o *timeout* estabelecido em 5000 ciclos equivale a 5000 ms, ou o mesmo que 5 s.

Depois de inicializado, o temporizador nomeado *debug_timer* é acompanhado periodicamente por meio de um laço infinito e, toda vez que a contagem de 5 s

for atingida, uma função de depuração é chamada para executar qualquer tarefa que seja necessária.

Não é redundante reforçar que a utilização desta implementação só é recomendada para o controle de intervalos que não interfiram em condições críticas do sistema, sendo mais interessante, nesses casos, o uso dos periféricos dedicados e seus mecanismos de interrupção de *hardware* que atribuem maior prioridade aos estouros de contagem de tempo.

4.3.4 Acionamentos de Motores CC por Modulação PWM

Com o intuito de estabelecer um acionamento eficiente aos motores do robô, foi decidido pelo uso de sinais *PWM* como acionamento das chaves que compõem o circuito em ponte, responsável por fornecer energia à carga. Para a geração desse tipo de sinal em microcontroladores faz-se uso dos periféricos dedicados a temporizações, denominados *timers*, e caracterizados pela contagem progressiva de ciclos de tempo, acumulando o índice de contagem até atingir determinada condição de reinício. Em microcontroladores modernos, os *timers* possuem modos de operação nativos específicos que disponibilizam nos pinos de saída sinais com características de razão cíclica controladas e frequências adequadas às especificações definidas pelo usuário.

Para habilitar a funcionalidade de *PWM* em um dos *timers*, basta selecionar o canal desejado na barra lateral da aba *Pinout* no projeto criado pelo *STM32CubeMX*, atribuindo a ele o modo de operação *PWM Generation CHx*. Para a aplicação de acionamento, é possível também fazer uso de um mecanismo específico da interface *TIM1* do *STM32F103C8TX*, que permite a geração de dois sinais complementares inversos e a configuração destes como atuadores para o mecanismo de segurança que evita o acionamento de duas chaves pertencentes a um mesmo braço vertical da ponte, que pode ocasionar um curto no barramento de alimentação. Para habilitar esse modo de operação, a opção correta passa a ser *PWM Generation CHx CHxN*, que é encontrada para os canais 1, 2 e 3 do *TIM1*, e direciona o sinal normal e o sinal inverso para dois pinos de saída distintos do microcontrolador.

Como existem duas pontes de controle no circuito, será necessário o uso de dois canais simultâneos para a geração de sinais, nesse caso, os canais 1 e 2. A utilização desta configuração nativa também é interessante, pois leva em conta situações como o tempo morto necessário entre a troca de estados dos dois sinais, já que, no período de transição, uma chave pode ser acionada indevidamente e causar o curto indesejado.

Depois de definir o comportamento geral da interface, o próximo ponto de interesse diz respeito à frequência do sinal *PWM* gerado, que deve ser escolhida seguindo alguns critérios: para aplicações com motores de corrente contínua, a inércia

mecânica tem o efeito de causar uma menor dissipação da energia entre as rotações, funcionando como um filtro para frequências mais altas e fazendo com que a frequência necessária para o acionamento desses elementos não precise ultrapassar a ordem de alguns kHz. Outro fator importante na seleção da frequência de acionamento diz respeito ao ruído audível causado por frequências entre os 20 Hz e os 20 kHz, mais agudas e incômodas quanto mais altas elas forem. Deste modo, unindo essas definições, para este estudo foi escolhida a frequência de 1 kHz como base para o sinal PWM de acionamento, relativamente baixa na faixa audível, porém, com uma margem suficiente em relação à taxa de rotação do motor e de fácil implementação em qualquer dispositivo microcontrolado.

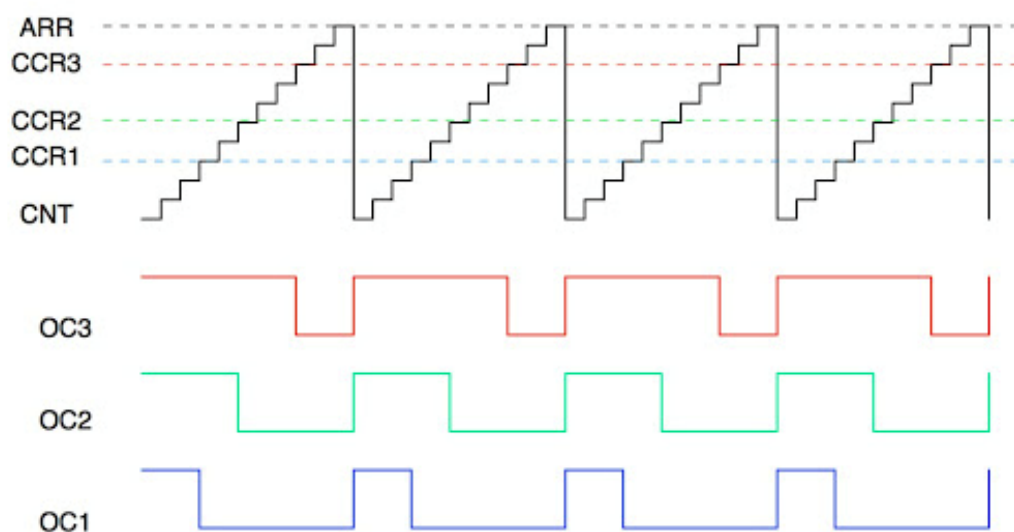
Para atingir essa frequência do sinal, dois parâmetros precisam ser balanceados: o *prescaler* do *clock* e o limite de contagem estabelecido para o *timer*. Como mencionado anteriormente, o ajuste do *prescaler* diz respeito a uma derivação do *clock* de entrada do periférico que permite uma nova taxa de operação. Para esta interface, foi designada uma redução de 8 vezes sobre o *clock* global, obtendo uma frequência resultante de 8 MHz. Essa atualização, na definição do *clock*, reflete diretamente no ajuste necessário para o limite de contagem, já que cada incremento de contador passa a equivaler a um intervalo maior, quanto menor for a frequência de atualização.

Ainda, um sinal de *PWM* prevê que em parte definida do tempo a saída apresentará valor alto, enquanto no tempo restante, o nível lógico ocupado pela saída será baixo, com essa proporção definida pela chamada razão cíclica. Em termos de configuração essa proporção é convertida no parâmetro *Pulse*, que dita por quantos ciclos do total (definido no *ARR*) o sinal de saída manterá nível de tensão alto antes de transitar para o nível baixo. Como essa variável será alterada ao longo da execução do código, sua inicialização não é crítica, entretanto, para exemplificar, no canal 1 foi escolhido uma quantidade de 5.000 ciclos para o *Pulse*, que equivale a uma razão cíclica de 0,625 em relação aos 8.000 ciclos totais do período. Atribuir esse diferentes valores de comparação reflete no comportamento também previsto na Figura 33, que representa três diferentes tipos de saída, dependendo do valor de *Pulse* escolhido em cada caso, representado pelas variáveis *OCX*.

Pensando no modo de funcionamento de um *PWM*, o limite de contagem representa o fim de um período para o sinal, momento em o ciclo reinicia e o sinal transita novamente para nível lógico alto, conforme evidenciado na Figura 33. Esse limite pode ser configurado por meio do parâmetro *AutoReload Register (ARR)*, atribuindo o número de ciclos que precisam ser computados até que passe o intervalo de tempo desejado como, por exemplo, se tomar uma frequência de 8 MHz, que equivale a um período de 125 ns, será necessária a passagem de 8.000 ciclos para atingir um intervalo de tempo igual a 1 ms, período que reflete em uma frequência final de 1 kHz,

selecionada neste estudo.

Figura 33 – Exemplos de sinais de saída PWM para diferentes níveis de comparação



Fonte: <www.micromouseonline.com>.

Vale lembrar que essa descrição leva em conta a utilização do *PWM* do microcontrolador em seu modo de contagem crescente e unidirecional, que inicia sempre em 0 e segue até o valor máximo, momento em que reinicia o processo todo e repete a operação anterior. Modos alternativos permitem contagem decrescentes ou mesmo até mesmo crescentes e decrescentes, onde o valor do contador é incrementado até o seu máximo e, quando o atinge, passa a decrescer, lembrando uma onda triangular. O uso de diferentes tipos de contagem reflete em diferentes comportamentos de saída e distribuição dos tempos de alta e baixa no sinal *PWM*.

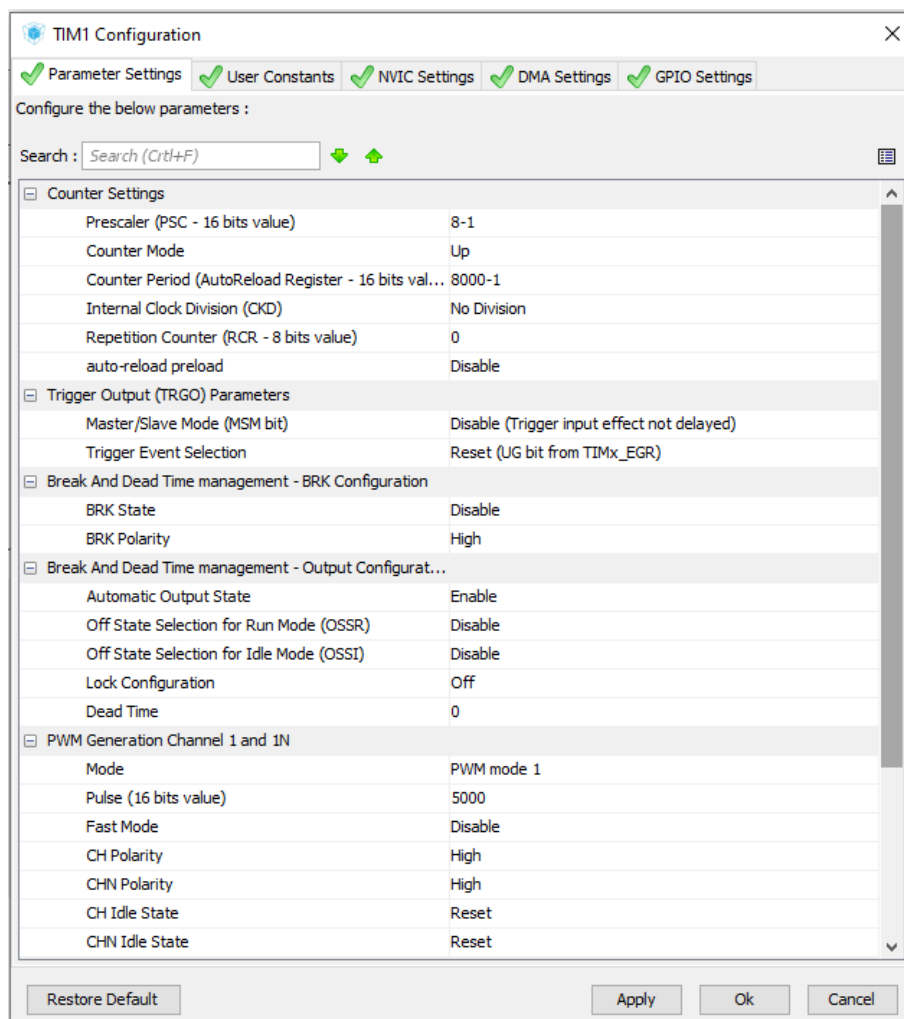
A interferência sobre todas essas definições pode ser feita por meio do menu de configuração da interface *TIM1* que, depois de selecionar o modo de operação com sinais complementares, oferece uma série de condições específicas para o ajuste desta condição, que podem ser visualizadas na Figura 34.

A partir disso é possível estabelecer a rotina em *software* para o controle do acionamento desejado, por meio das condições estabelecidas de início e fim da geração de sinais *PWM* que atuam sobre o circuito. Toda vez que a rotina de controle é chamada, ela deve levar em conta a direção de deslocamento e a velocidade que deve ser atingida.

Para um motor de corrente contínua, a velocidade de rotação é diretamente proporcional ao nível de tensão aplicado sobre seus terminais, fator que, em termos de controle de *PWM*, é reflexo da seleção do parâmetro *Pulse*, que controla a razão cíclica do sinal e, quanto maior seu valor, maior a tensão média de saída. Outro detalhe importante é entender que o motor deve ser alimentado com uma tensão mínima, já

que opera na faixa entre 3 e 6 V, o que reflete também numa razão cíclica mínima que, para uma tensão de alimentação de 5 V, equivale a 0,6. Desse modo, na rotina de atualização deve haver a garantia de um valor de pelo menos 4.800 ciclos para o *Pulse*, evitando atingir condições inviáveis de operação.

Figura 34 – Janela de configuração dos parâmetros para o Timer TIM1



Fonte: Adaptado de STM32CubeMX v4.26.1.

Em termos de direção, sabe-se que a inversão no sentido da corrente que percorre os terminais do motor resulta em direções de rotação distintas. A tradução disso no controle da ponte-H é o acionamento de um conjunto diferente de chaves, feito pela utilização dos sinais complementares PWM e PWM_N , utilizados alternadamente e capazes de gerar essa inversão de polaridade de alimentação ao motor, conforme descrito em tópicos anteriores.

O Código A.4 estabelece um exemplo de implementação para função que atualiza a direção de deslocamento do motor, chamada sempre que necessitar mudar o comportamento do robô, prevendo a movimentação em quatro modos distintos e uma

condição de parada, com o desligamento dos sinais de acionamento.

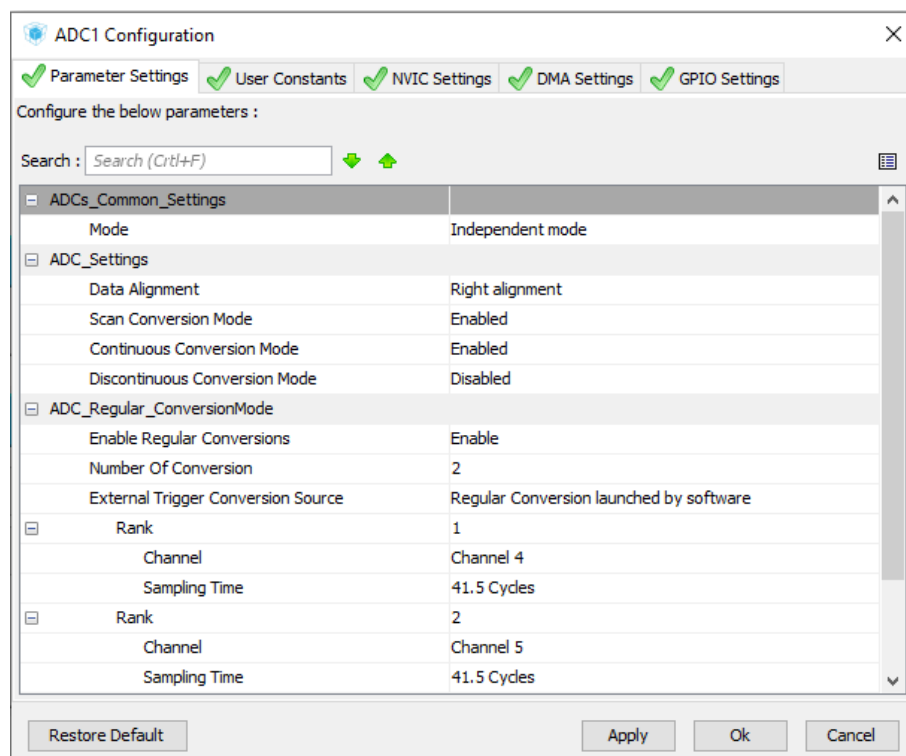
Como é perceptível no código, para o robô avançar, os sinais de ambos os canais de *PWM* são acionados, resultando na definição do mesmo sentido de alimentação em ambas as pontes e, consequentemente, nos quatro motores. Caso deseje retroceder, a configuração se inverte em ambas as pontes e o sinal PWM_N passa a operar ativamente em ambos os canais, refletindo na inversão de rotação de todos os motores. Para situações em que a mudança de direção exige que o eixo da estrutura do motor seja manipulado, como uma conversão à direita ou à esquerda, o lado para o qual se deseja virar tem sua rotação reduzida, enquanto o outro é mantido em maior velocidade, fazendo com que a disparidade de velocidades naturalmente tire o carrinho de seu eixo inicial. A última condição prevista diz respeito ao estado em que o carrinho deixa de se movimentar, quando ambos os sinais de controle são desligados.

Um último nível de controle adicionado neste trecho do programa leva em conta a ideia de manter um poder de seleção sobre a velocidade atingida pelo sistema: como discutido nos tópicos anteriores, quanto maior a razão cíclica, maior a velocidade, entretanto, como não foi prevista a leitura da velocidade instantânea de rotação do motor, que permite um controle dinâmico ao mesmo, esse fator de mudança é feito por uma variável *speed*, prevista para alcançar valores entre 0 e 2, o que permitiria razão cíclica entre 0,625 e 0,875, garantindo uma faixa de variações de velocidade.

4.3.5 Leituras dos canais do ADC

Como discutido anteriormente, com foco no prolongamento da vida útil das baterias de alimentação, o conversor analógico-digital interno ao *STM32F103* será utilizado para acompanhar periodicamente o valor de tensão sobre as fontes e, para isso, deve ser ter o comportamento de leitura de suas entradas configurado na interface do *STM32CubeMX*. Na aba *Configuration*, já mencionada, é possível acessar menus de configuração para os periféricos do sistema que foram ativados pelo usuário. A ativação do *ADC* ocorre ao selecionar algum dos canais de entrada ao atribuí-los aos pinos de entrada e saída. Para este projeto, os pinos configurados como entrada analógica foram o *PA4*, vinculado ao canal de número 4 do *ADC1*, e o *PA5*, quinto canal da mesma interface.

Ao selecionar o menu, o usuário passa a ter acesso a uma janela contendo os parâmetros disponíveis para a personalização do periférico, que pode ser visualizada na Figura 35, contendo as configurações definidas para a aplicação especificada neste projeto.

Figura 35 – Janela de configuração dos parâmetros para o conversor ADC interno

Fonte: Adaptado de STM32CubeMX v4.26.1.

Basicamente, a sequência de configurações disponível para o *ADC* consiste em dois grupos principais: os parâmetros que selecionam o modo de operação do periférico e os que definem as características das aquisições realizadas. Alguns destes parâmetros têm seus estados interdependentes fazendo que, ao aplicar determinada configuração em um deles, as opções para os parâmetros do outro sejam modificadas. Um bom exemplo disso é a relação entre o *Scan Conversion Mode* e o *Number of Conversions*: o modo de escaneamento leva em conta o número de conversões que devem ser realizadas, sendo ativado automaticamente quando o sistema necessitar de duas leituras ou mais e desativado caso apenas um canal seja utilizado. Outro exemplo, são o *Continuous Conversion Mode* e o *Discontinuous Conversion Mode* que, como sugerem seus nomes, são mutuamente exclusivos.

Outras alterações podem ser realizadas, como a especificação do *Data alignment*, que diz respeito à quantidade efetiva de *bits* utilizados para armazenar a variável de leitura, necessitando de dois registradores para armazenar valores de ordem maior que 8 *bits*. Por padrão, o *STM32CubeMX* seleciona o alinhamento à direita, que utiliza todos os *bits* da resolução suportada pelo *ADC* e depende dos dois registradores de destino. Também neste menu é possível estabelecer um gatilho para a conversão dos valores, por meio do parâmetro *External Trigger Conversion Source*, que pode ter origem no *software* da aplicação ou em um dos *timers* do sistema. Mesmo

que o sistema faça a leitura do nível de tensão periodicamente, é possível garantir que a conversão desses valores pelo *ADC* só seja realizada quando houver uma requisição, reduzindo o tempo que o conversor processa as informações e uma sobrecarga do periférico.

Ao configurar o *Number of Conversions* como 2, surge a possibilidade de estabelecer uma ordem de prioridade para os canais da interface utilizada, importante para a leitura das informações pelo programa, que será feita sequencialmente. Para o contexto de uso, definido pelo projeto, foi especificado que o pedido para realização de uma conversão deverá partir do *software* e que a ordem de verificação dos canais será crescente em relação aos seus índices, atribuindo o *Rank 1* para o canal 4 e o *Rank 2* para o canal 5.

Passando para a parte de implementação em *software*, embora o *ADC* possa fazer uso de uma interrupção dedicada para gerenciar as leituras, neste projeto foi escolhido o modo de aquisição pelo método de *polling*, que atribui ao programa a tarefa de requisitar que uma conversão seja feita, condição compatível com verificações sem critérios temporais estreitos, e menores prioridades de execução. O controle de início e fim de cada um dos ciclos de leitura pode ser feito por meio da interação do programa com uma das instâncias de interface do *ADC*, gerada a partir da configuração do periférico, realizada anteriormente, que garante o acesso às funções específicas de manipulação de seu funcionamento. A operação simples de leitura utilizada para a verificação dos sinais de tensão pode ser acompanhada no Código A.5.

O valor definido como *Sampling Time* pode causar certa confusão: embora pareça estar relacionado com a frequência de aquisição do sinal, na realidade essa informação dita o tempo de carga para o capacitor responsável por reter o nível da tensão fornecido na entrada. Um método comum de leitura de dados para um conversor analógico-digital é o chamado *Sample and Hold*, baseado na definição que após a identificação do nível do sinal, a informação deve ser retida, inalterada, até que a conversão se complete e uma nova leitura seja realizada. Esse funcionamento depende diretamente da carga de um capacitor, alimentado com a diferença de potencial e com a função de armazenar o valor identificado até o momento de ser convertido pelo *ADC*. Desse modo, o *Sampling Time* está diretamente vinculado ao número limite de ciclos para a tolerância de carga deste capacitor, garantindo que, mesmo para entradas de alta impedância, que demoram mais tempo para carregá-lo, o capacitor terá tempo suficiente para atingir o nível da tensão de entrada, evitando que valores incorretos sejam lidos no meio do processo de carregamento. É importante lembrar que, quando utilizando todos os 12 *bits* de precisão do *ADC*, outros 12 ciclos serão necessários para o processamento efetivo da conversão do valor adquirido, independentemente do *Sampling Time* selecionado. Caso a taxa de aquisição não seja um fator crítico, é

recomendado manter esse parâmetro no maior intervalo possível.

O primeiro ponto importante visualizado nessa rotina é a identificação dos passos que levam à leitura de cada canal, feita de maneira sequencial. O processo é iniciado com a chamada da função *HAL_ADC_Start()*, que recebe como parâmetro um ponteiro com a estrutura da instância da interface selecionada e que é responsável por notificar ao periférico o começo da operação de acesso a suas informações. Logo após este anúncio é emitida a requisição da conversão, por meio da função *HAL_ADC_PollForConversion()*, que estabelece um tempo de espera para que o *ADC* realize a conversão e processe o dado requisitado para, em seguida, atribuir o resultado a variável de destino adequada, utilizando a função *HAL_ADC_GetValue()*. Como mencionado no processo de configuração deste periférico, a sequência de leitura dos canais segue a ordem dos *Ranks* atribuídos aos canais de interesse.

O resultado da conversão vindo do *ADC* é um valor proporcional que identifica a relação entre o valor máximo de referência do conversor, neste caso 3,3 V, e a tensão de entrada, posicionando esse resultado em uma régua que vai de 0, que equivale a 0 V a $2^N - 1$ (com *N* representando a resolução em *bits* do conversor), que equivale a 3,3 V. Desse modo, a identificação dessa informação ainda não traduz a dimensão real do valor lido, necessitando um ajuste para obter o significado real atribuído a essa proporção. De maneira direta, para identificar o nível de tensão V_{in} que foi aplicado à entrada, basta realizar uma regra de três simples, na qual a proporção de referência é a própria relação entre o índice máximo de contagem e o nível de tensão que ele representa. Simplificando, a interpretação do valor convertido pelo *ADC* na leitura segue a Equação 4.5.

$$V_{in} = \frac{ADC_{value}}{2^N - 1} V_{ref} \quad (4.5)$$

Com esse ajuste, a variável já consegue representar o nível de tensão aplicado à entrada do *ADC*, entretanto, como visto na subseção 4.2.1.2, o valor real lido diretamente das baterias precisou passar por um estágio intermediário de compensação para que fosse suportado pelos limites de tensão do periférico, necessitando da aplicação de um fator de multiplicação que permita a obtenção do valor real de cada ponto de leitura. Para o primeiro ponto, que identifica a tensão de uma única célula, o fator de multiplicação deve ser de 3 vezes, compensando a relação estabelecida pelos resistores selecionados no amplificador diferencial relacionado a ele, enquanto que o fator de multiplicação para o segundo ponto de leitura deve ser igual a 1,5 vezes. Como o segundo ponto mede a tensão sobre as duas baterias, basta subtrair o resultado obtido no primeiro ponto para encontrar a tensão individual da outra célula.

A leitura adequada dessas informações é de extrema importância para o

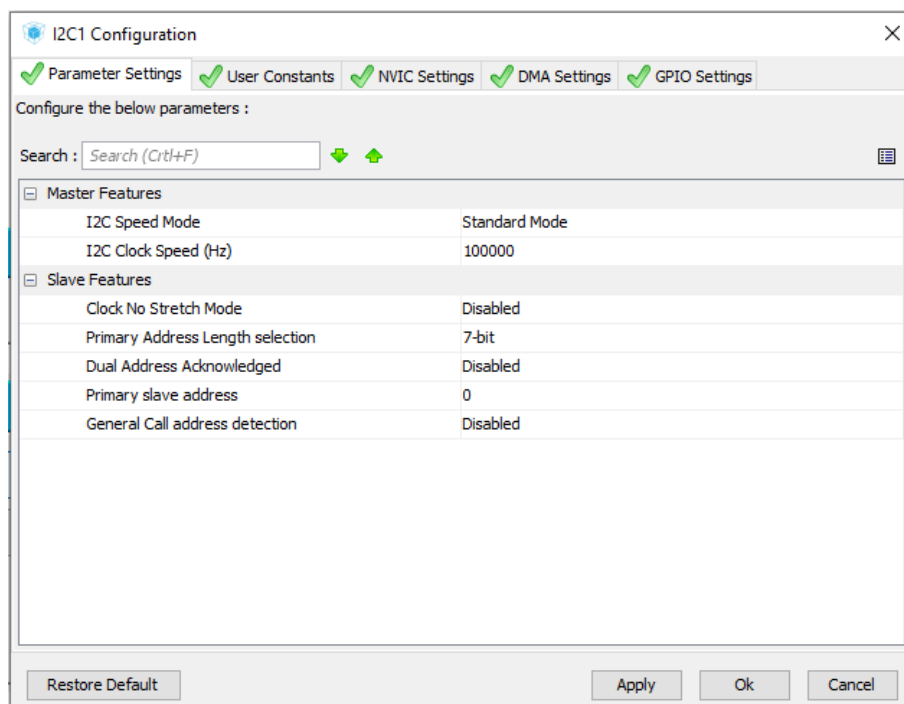
sistema já que estes valores serão o critério de identificação das condições adequadas de operação para as baterias que alimentam o robô. Como mencionado anteriormente, as baterias empregadas neste estudo possuem um nível mínimo de tensão para operar em segurança, necessitando de um processo de recarga para sua reutilização. Com os valores lidos, o microcontrolador fica responsável por avaliar se o valor crítico foi atingido e, em caso afirmativo, deve avisar ao usuário que o estado da bateria inspira atenção. Para essa notificação o *hardware* conta com um *LED* dedicado, que se acende assim que o valor de tensão sobre qualquer uma das baterias atingir valor mínimo, permitindo ao usuário entender o momento certo para recarregar os elementos armazenadores antes de danificá-los permanentemente.

4.3.6 Comunicação pela interface I^2C

Depois de condicionar o barramento I^2C em termos de *hardware*, com a adição dos resistores de *pull-up* e conexão correta dos dispositivos, o trabalho de interação com essa interface fica bastante simplificado, dependendo apenas do ajuste de alguns parâmetros pelo *STM32CubeMX*.

A principal alteração que pode ser feita diz respeito ao *clock*, utilizado como sinal de sincronia para as transações no barramento: o periférico I^2C presente neste microcontrolador permite a seleção entre dois modos de velocidade, um *Standard*, que permite um *baud rate* de até 100 kbps e um modo *Fast*, que ajusta as condições do sistema para permitir velocidades de até 400 kbps. A escolha dessa velocidade de transmissão depende basicamente da capacidade dos dispositivos conectados em se comunicar nessas condições e, como neste projeto ambos os módulos que utilizam a conexão I^2C possuem suporte para velocidades maiores que 100 kHz, a utilização do modo *Fast* é permitida. É possível também selecionar o tamanho da variável de endereçamento de dispositivos, que pode ter 7 ou 10 *bits* de comprimento. Algumas outras informações podem ser ajustadas, como evidencia a Figura 36.

O desenvolvimento de *software* para os módulos com comunicação I^2C passou por um processo diferente das demais implementações do sistema desenvolvido neste projeto, já que fez uso de bibliotecas de terceiros para a interação com estes dispositivos. Por meio de pesquisas em sites de buscas, foi identificada a existência de duas bibliotecas distintas, que permitem a interação com o *BMP180* e o *MPU6050*, circuito integrado interno ao módulo *GY-521*, que realiza o gerenciamento dos acessos aos sensores e também da comunicação com dispositivos externos. O filtro na busca pelas bibliotecas garantiu que estas fossem compatíveis com a plataforma utilizada, principalmente que se baseasse em rotinas de programa destinadas ao *STM32* como base para as funções de interação com *hardware*, facilitando a integração entre os conjuntos por não depender de grandes ajustes.

Figura 36 – Janela de configuração dos parâmetros para a interface I²C

Fonte: Adaptado de STM32CubeMX v4.26.1.

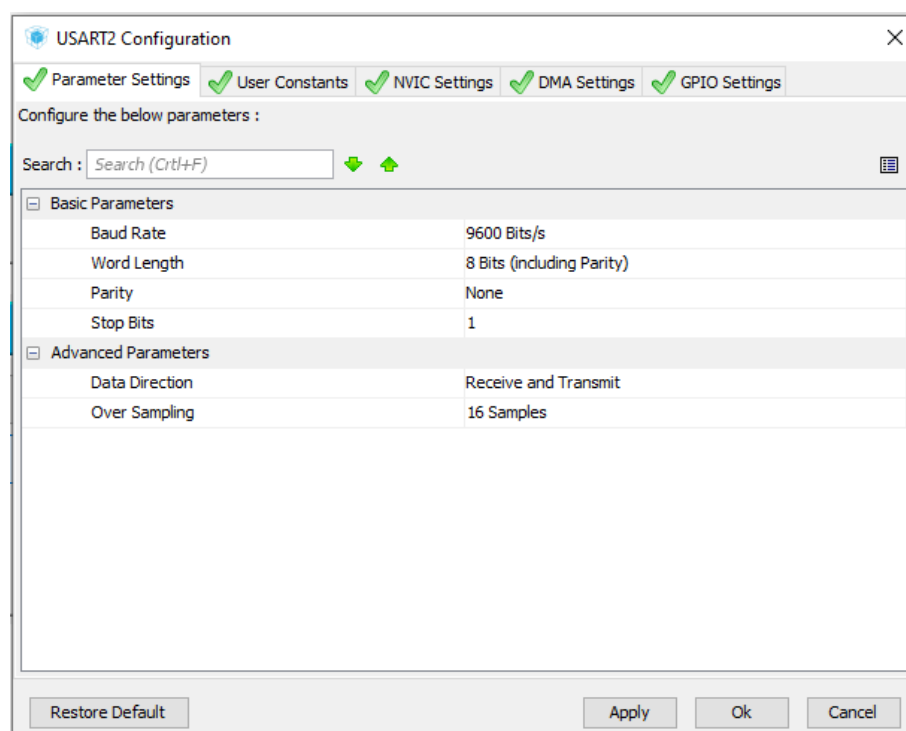
O código direcionado ao módulo *BMP180*, foi disponibilizado na plataforma *GitHub* por meio de um *Gist* e pertence a Gheorghe Girjau, desenvolvedor de soluções para sistemas embarcados (GIRJAU, 2019). A implementação criada por Gheorghe é capaz de utilizar o barramento para inicializar o sensor e ler os parâmetros de calibração dos quais depende a conversão das informações adquiridas e, depois de devidamente inicializado, requisitar periodicamente ao sensor que faça a aquisição dos dados e os retorne ao microcontrolador que, por meio de funções também estabelecidas na biblioteca, consegue transformar o dado cru em informações legíveis. Esse processo todo é bastante complexo e o uso de uma biblioteca dedicada permite uma interação facilitada com o dispositivo, sem se preocupar diretamente com o cálculo realizado, apenas utilizando o resultado como base para decisões dentro da aplicação.

De mesmo modo, o código direcionado ao módulo *GY-521* também foi encontrado na plataforma *GitHub*, pertencente ao desenvolvedor russo Bulanov Konstantin (KONSTANTIN, 2020). Em seu código, Konstantin permite a inicialização do dispositivo e também a conversão das informações, passando inclusive por um processo de adequação das leituras, por meio do uso de um filtro de Kalman. Um detalhe interessante é que, na página inicial do projeto, contendo as descrições gerais da implementação, Konstantin evidencia o uso da plataforma *STM32CubeMX* como base para a criação do projeto, seguindo o mesmo processo descrito neste estudo.

4.3.7 Comunicação pela interface *UART*

Para utilizar a interface USART pertencente ao *STM32F103C8TX*, após selecionar corretamente os pinos desejados e atribuir o modo de funcionamento para assíncrono, utilizando a barra lateral da aba *Pinout*, basta realizar alguns ajustes por meio do menu encontrado na aba *Configuration* do *STM32CubeMX*. Em termos gerais, os parâmetros que definem a conexão *UART* são seu *baud rate*, escolhido de modo que seja compatível com o dispositivo externo ao qual deve comunicar-se, e suas condições de construção da informação, com o tamanho da estrutura mínima de transmissão de dados e seus comportamentos em relação à paridade e *bit* de parada. Para facilitar a visualização, a Figura 37 mostra os parâmetros disponíveis no menu de configuração da *USART2*.

Figura 37 – Janela de configuração dos parâmetros para a interface *UART*



Fonte: Adaptado de STM32CubeMX v4.26.1.

Neste projeto foram selecionadas as configurações mais tradicionais relacionadas a interfaces *UART*, com *baud rate* de 9.600 e configuração de palavra na sequência *8N1*, contendo 8 *bits* de informação, 1 *bit* de parada e a ausência de paridade. Essa configuração foi escolhida também por ser compatível com o comportamento padrão do módulo *HC-05*, que pode ser configurado para outros modos de operação, entretanto, depende de rotinas auxiliares de condicionamento.

A biblioteca que prevê o recebimento de informações pelo protocolo *UART*, obtida da documentação da *ST Microelectronics*, mesmo capaz de receber pacotes com

dimensão superior, ainda é limitada pelo comportamento do *hardware* de processar apenas um *byte* de informação por vez. Desse modo, foi decidido que a recepção dos pacotes seria feita de modo a concordar com a limitação do *hardware*, processando também o recebimento de apenas um *byte* por vez. Com isso, a preocupação desse gerenciamento da informação recebida passou a ser sobre a identificação de início e de fim de um pacote, já que este não possui tamanho fixo e pode sofrer interferências externas no meio de uma recepção.

O primeiro passo na solução deste problema foi definir que toda informação que circula pelo barramento da *UART* seja codificado nos parâmetros *ASCII*, mesmo os valores inteiros, que devem ser convertidos apropriadamente. Com isso, as informações que transitam podem assumir apenas os valores correspondentes a letras e números nessa codificação, permitindo que caracteres especiais sejam usados como limitadores de pacote. Para cumprir a tarefa, o caractere '@' foi determinado para sinalizar o início de uma transmissão de dados e o caractere '!' passou a sinalizar o fim de um pacote.

A segunda definição estabelecida diz respeito aos pacotes recebidos pelo sistema, contendo informações de controle, que permite utilizar apenas valores em notação hexadecimal para relacionar as diferentes interações de dispositivos externos com o robô. Ao definir essa notação, significa que cada *byte* lido como um caractere representa efetivamente apenas um *nibble*, ou 4 *bits* de informação, de um *byte* em uma variável de tipo inteiro na linguagem C. De maneira mais clara, um algarismo que é representado por um *byte* em *ASCII*, representa apenas metade do *byte* em notação hexadecimal. Com isso, para reconstruir os comandos em estruturas internas, seria necessário converter o valor em *ASCII* para um inteiro e posicioná-lo adequadamente no vetor de *bytes* que representa o *buffer* de recebimento de informações.

Com essas definições de comportamento é possível estabelecer uma rotina de interpretação das informações recebidas, reconstruindo o pacote enviado por um dispositivo externo, que é decomposto em unidades menores (*bytes*), posteriormente identificando o comando e realizando a tarefa relacionada a ele. No Código A.6 uma sequência de funções compreende a leitura do *byte* na interface e a reconstrução do pacote até a identificação do caractere de parada. O modo de recepção da *UART* foi estabelecido em *polling*, sem utilizar a interrupção de recepção disponível para o periférico.

Após as definições de reconhecimento do pacote, é possível estabelecer um protocolo básico para uma série de comandos e parâmetros derivados, permitindo o controle das funções internas do sistema. Em um exemplo de implementação, o primeiro *byte* de informação identifica o comando desejado, o segundo sinaliza qual parâmetro de funcionamento sofrerá interferência e os demais são liberados para compreender informações adicionais sobre a requisição. É importante lembrar que

essa construção é bastante maleável, dependendo apenas de que as informações sejam compostas em *bytes* completos (ex: '0x01' no lugar de '0x1') e que os caracteres de início e fim sejam incluídos na sequência. Na Tabela 7, estão descritos alguns comandos suportados pelo sistema e os parâmetros associados a cada um.

Tabela 7 – Protocolo de comunicação com os comandos para o sistema

Comandos	Parâmetros A	Parâmetros B
0x01 - Direção de Movimento	0x01 - Operação 0x02 - Direção	0x00 - Modo Normal 0x01 - Modo Seguidor de Linha 0x01 - Avançar 0x02 - Retroceder 0x03 - Virar à esquerda 0x04 - Virar à direita 0x05 - Parar
0x02 - Rotina dos <i>LEDs</i>	0x01 - Operação 0x02 - Seleção de rotina	0x00 - <i>LEDs</i> apagados 0x01 - <i>LEDs</i> acesos 0x00 - Rotina padrão 0x01 - Rotina alternativa A 0x02 - Rotina alternativa B
0x03 - Controle do <i>Buzzer</i>	0x01 - Operação 0x02 - Volume	0x00 - Som desligado 0x01 - Som ligado 0x00 a 0x99 - Controle volume (valores inteiros)
0x04 - Requisição para atualizar sensores	-	-

Fonte: Autoria Própria.

Na recepção, a identificação dos comandos também pode ocorrer de maneira simplificada, comparando as informações recebidas no *buffer* com os valores de referência definidos em protocolo, redirecionando os parâmetros necessários para atribuição das diferentes funções. No Código A.7, é identificável uma estrutura *switch*, responsável por direcionar as requisições às funções de reação correspondentes.

Para o envio de eventos, o processo utilizado pode ser o mesmo, com o uso dos caracteres limitadores de pacote e a instauração de códigos definidos para cada um dos eventos emitidos pelo sistema. No Tabela 8, um exemplo de protocolo que descreve a atualização dos estados dos sensores internos e permite a dispositivos externos disponibilizar essas informações para a leitura ou processamento adicional, conforme a necessidade do usuário.

Tabela 8 – Protocolo de comunicação com os eventos gerados pelo sistema

Eventos	Parâmetros (<i>strings</i>)
0x10 - Atualiza temperatura	0 a 100 - Valores inteiros
0x11 - Atualiza pressão	0 a 100 - Valores inteiros
0x12 - Atualiza altitude	0 a 100 - Valores inteiros
0x13 - Status da bateria	00 - Nível crítico 01 - Nível baixo 02 - Nível médio 03 - Nível alto

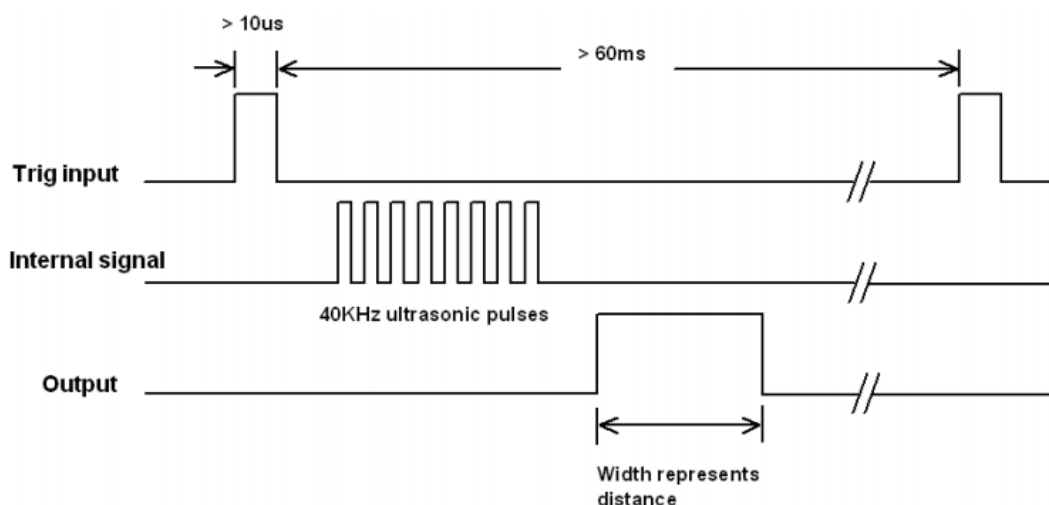
Fonte: Autoria Própria.

Como exemplo de uma utilização do protocolo de geração de eventos, o Código A.8 explicita a verificação das leituras do *ADC* para os valores de tensão nas células da bateria utilizada como alimentação do sistema, identificando os diferentes níveis de carga e sinalizando, por meio de um dos parâmetros definidos na tabela de protocolo, em qual dos patamares especificados encontra-se o nível da carga das baterias.

4.3.8 Leitura do sensor ultrassônico

Diferente dos outros módulos conectados a este *kit* didático, que utilizam conhecidos protocolos para transmitir os dados de leitura ao microcontrolador, o *HC-SR04*, medidor ultrassônico de distância, utiliza como meio de troca de informações as entradas e saídas comuns da plataforma de controle, além de obrigar que esta realize a interpretação dos dados obtidos. Desse modo, a comunicação com este dispositivo tem dependência direta em parâmetros temporais, refletidos em pulsos de comprimento determinado e tempos de espera específicos. Na Figura 38, é possível visualizar o diagrama temporal que descreve uma dessas leituras de distância, permitindo compreender melhor a interação entre os sistemas.

De maneira prática o *HC-SR04* não depende de qualquer sequência de inicialização, apenas de um disparo do pino *TRIG* para começar uma aquisição. Basicamente, para o microcontrolador deve enviar um pulso em nível *TTL* alto por pelo menos 10 μ s, indicando que deseja realizar a medição e, após receber o sinal, o módulo emite 8 pulsos ultrassônicos a 40 kHz, por meio do seu atuador, abrindo em seguida seu receptor para a identificação do retorno da onda mecânica. Ao receber a resposta dos oito ciclos, o módulo processa os resultados e transforma o valor final de leitura em um pulso de duração proporcional à distância encontrada, que deve ser disponibilizado no pino *ECHO*. Desse modo, ao receber a informação, o microcontrolador deve medir esse intervalo de duração e aplicar os coeficientes de proporção para decodificar a

Figura 38 – Diagrama de tempo para a leitura do módulo *HC-SR04*

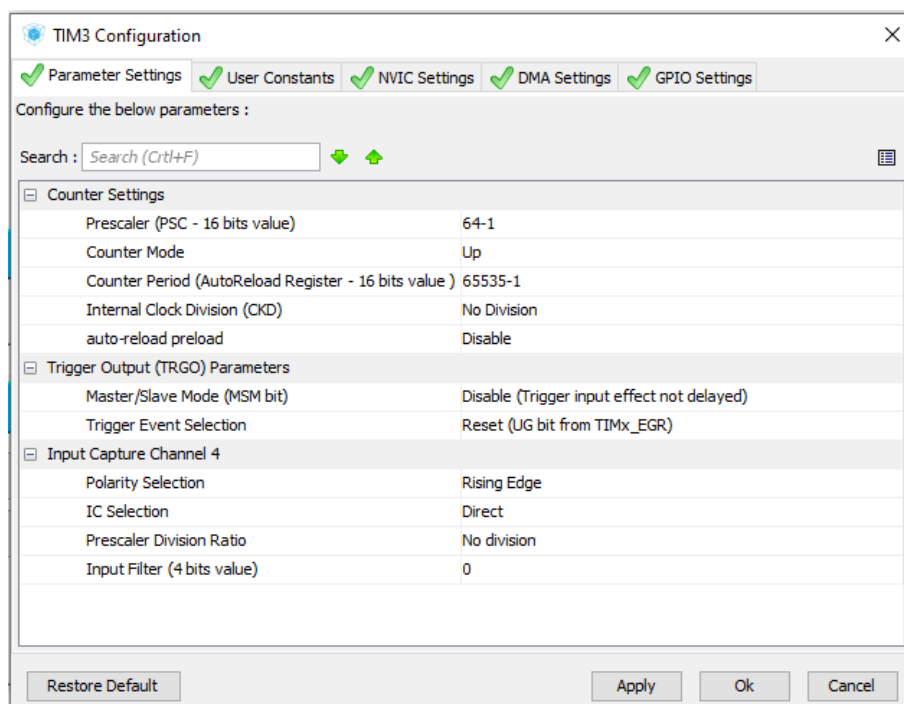
Fonte: (ANJITHA, 2016).

leitura (MORGAN, 2014).

Para o acionamento do pino *TRIG*, não existe segredo, basta temporizar a aplicação para que se escreva um nível lógico alto no pino correspondente e, ao atingir o limite de $10\mu s$, retorne o estado do pino a nível lógico baixo. Já para a leitura, o processo de identificação do resultado é feito de maneira mais complexa, dependendo de um tratamento rebuscado para medir a duração do pulso de resposta. Embora a temporização seja um parâmetro importante no gerenciamento na execução das instruções e rotinas internas, nesse casos ela é a base de sincronia e não a variável a ser identificada, como ocorre nesta situação e que acaba por dificultar a execução.

Focando inicialmente no pulso de ativação, a preocupação é direcionada a identificar uma maneira de contar períodos de tempo na escala de microssegundos, abaixo do mínimo suportado para a biblioteca padrão do *STM32*, que baseia suas interações em *hardware* na escala dos milissegundos. De forma natural, essa implementação vai depender de um dos periféricos do sistema dedicados à temporização, que consegue utilizar o *clock* global para estabelecer contagens personalizadas de períodos de tempo. Tendo isso em mente, o primeiro passo é estabelecer uma base de tempo em microssegundos, por meio da adequação de um *timer*, neste caso a interface *TIM3*, com as configurações relacionadas na Figura 39.

Consultando a Figura 31, é possível notar que a frequência que rege a operação dos *timers* do sistema está definida como equivalente ao próprio *clock* global, que atua em 64 MHz, obrigando a intervenção por meio do *prescaler* da interface para atingir uma derivação de frequência na faixa desejada. O valor de frequência que permite a ocorrência de um período de onda igual a $1\mu s$ é a de 1 MHz, 64 vezes menor

Figura 39 – Janela de configuração dos parâmetros para o Timer TIM3

Fonte: Adaptado de STM32CubeMX v4.26.1.

que o valor fornecido aos *timers*. De maneira lógica, o valor de *prescaler* a ser utilizado deve compreender essa proporção de 64 vezes, resultando na condição desejada de temporização. É importante lembrar que o *prescaler* é, na verdade, equivalente ao coeficiente de divisão do *clock* reduzido em uma unidade, por conta de parâmetros internos do processamento e pelos índices em linguagem *C* que começam a contar em 0.

Com a base de tempo definida, é possível estabelecer em *software* uma função que cria um laço de contagem e transforma os intervalos de microssegundos em números de ciclos na frequência de 1 MHz, como é apresentado no Código A.9. Toda vez que a função é chamada, zera o valor de contagem do *TIM3* e verifica a todo momento se o intervalo, recebido como parâmetro, já foi atingido. Embora nesse caso a aplicação não apresente condições críticas de funcionamento, é importante lembrar que esse tipo de rotina apenas "queima" processamento da unidade de controle, impedindo que outras operações sejam realizadas nesse período, podendo ser substituído por implementações que se baseiem em interrupções para aumentar a eficiência e a confiabilidade.

Como o estouro do *timer* não é um parâmetro relevante para a contagem de tempo estabelecida aqui, sendo apenas a variação dos ciclos utilizada para verificar a passagem do tempo, o valor atribuído ao *AutoReload Register* foi selecionado como o máximo permitido para a resolução de 16 *bits*, garantindo que qualquer período de

tempo entre 0 e 65.535 microssegundos possa ser configurado.

Avaliando as funcionalidades disponíveis para o *STM32* e tomando como base uma implementação proposta pelo blog *Controllers Tech*, foi identificada a possibilidade de utilização de um dos canais do *TIM3* em modo de captura, permitindo a identificação do início do pulso no pino *ECHO* e o acompanhamento de sua duração, que identificaria o tempo de resposta em microssegundos. Para configurar este modo, basta selecionar a opção *Input Capture Direct Mode* para um dos canais do *timer*, através do menu lateral da aba *Pinout* no *STM32CubeMX*. É importante também habilitar a interrupção global do *TIM3*, por meio da aba *NVIC_Settings*, no menu do *TIM3* da aba *Configuration*. As funções que serão responsáveis por realizar as medições estão descritas no Código A.10 e foram baseadas na implementação proposta em CONTROLLERS TECH (2019), dedicada à plataforma *STM32* e interessante por seu comportamento voltado a interrupções.

Na primeira função, que inicia a leitura, o pino *TRIG* é inicializado em nível lógico baixo, para depois ser habilitado por cerca de 10 microssegundos. Após o fim deste pulso de ativação, a interrupção de captura é instanciada e a entrada conectada ao *ECHO* passa a aguardar uma transição de nível para identificar o início do pulso de resposta. Ao identificar a primeira transição, o tratamento da interrupção verifica o índice da contagem no *timer* para aquele instante, armazenando-o para posterior comparação, e configura a captura para que passe a aguardar uma transição de sinal em direção o nível lógico baixo. Quando a segunda transição for identificada, é lido o novo índice do *timer* e calculado o número de ciclos decorridos, que expressará um intervalo de tempo em microssegundos, já que cada ciclo do *TIM3* representa 1 μ s.

Para identificar a distância real em relação ao objeto, é preciso converter a medida do pulso, o qual simboliza o tempo entre a emissão do sinal ultrassônico pelo módulo e seu reflexo no receptor, utilizando valores físicos das ondas sonoras como coeficientes. Pensando que uma onda sonora propaga-se a aproximadamente 340 m/s, e que o tempo lido leva em conta os trajetos de ida e volta dessa onda, para calcular a distância d em centímetros percorrida num intervalo de $x/2$ microssegundos, é possível utilizar a equação descrita na Equação 4.6.

$$d(cm) = \frac{x * 10^{-6}}{2} * 340 * 100 = 0,017 * x \quad (4.6)$$

No processo de transcrição para a linguagem de programação, ocorreu uma compensação no coeficiente que multiplica x , para evitar que ocorram operações de divisão por números com ponto flutuante.

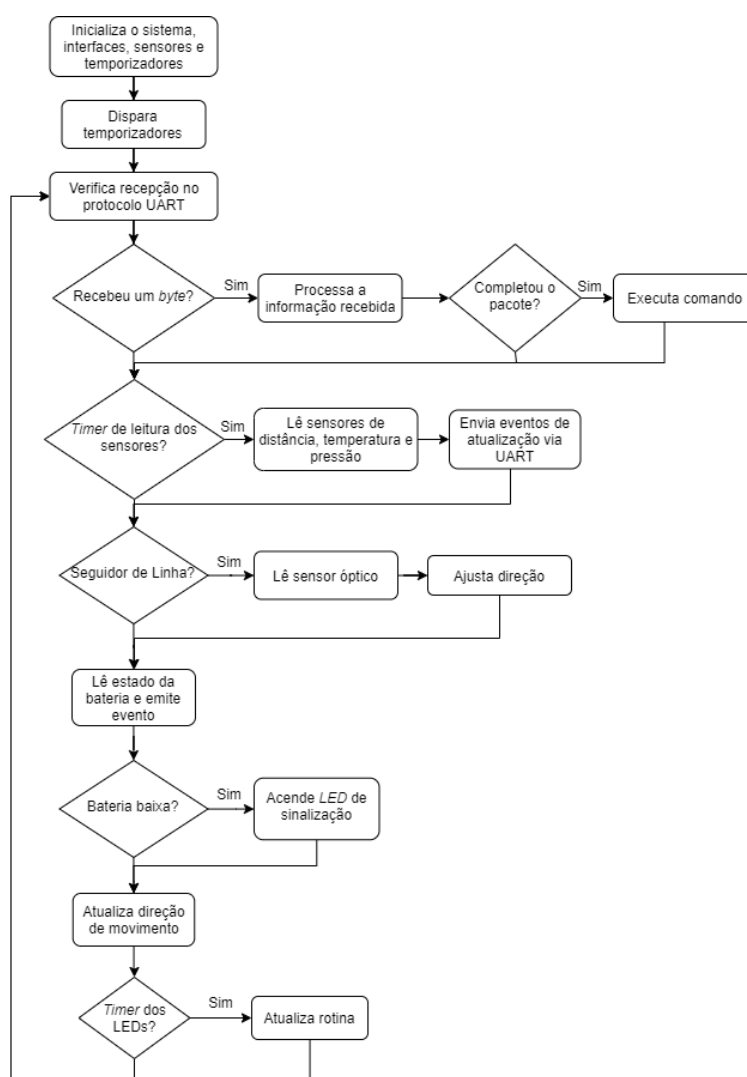
Outro fator importante que dever ser considerado para um melhor aproveitamento das funções deste módulo e pode ser visualizado na Figura 38, é o tempo

de recuperação do circuito até que possa ser realizada uma nova leitura, evitando que a segunda medição sofra interferências caso a primeira não tenha sido concluída. Verificado em sua folha de dados, o intervalo mínimo recomendado é de 60 ms entre duas leituras consecutivas, garantindo a aquisição correta dos dados.

4.3.9 Gravação do Programa e Cuidados Adicionais

Como resultado da série de configurações realizadas durante o processo de condicionamento de cada componente, surge um programa completo que reflete as interações do sistema com o ambiente em decisões simples, que resultam em atuações sobre o próprio sistema e informações relevantes ao usuário. Para visualizar melhor a sequência de execução da aplicação desenvolvida, a Figura 40 representa um fluxograma de abstração das principais ações tomadas no laço de execução do programa construído ao longo da execução do projeto do *kit*.

Figura 40 – Fluxograma da rotina de execução do *software* da aplicação



Fonte: (Autoria Própria, 2020).

Depois de configurar corretamente todo o *software* responsável por gerenciar o funcionamento de todos os elementos, para testar a implementação, é necessário carregar o programa para a memória do microcontrolador. Para essa tarefa, é possível utilizar diferentes tipos de ferramenta.

Como mencionado anteriormente, a utilização de uma *IDE* específica para o desenvolvimento do *software* não é obrigatória, entretanto, algumas delas possuem mecanismos nativos de integração com a plataforma utilizada. Um exemplo disso, é a ferramenta *Atollic TrueSTUDIO for STM32*, disponível para sistemas operacionais *Windows* e *Linux*, indicada pelo fabricante e uma das interfaces selecionáveis para integração no menu de configurações de destino do projeto no *STM32CubeMX*. Como diferencial, o *TrueSTUDIO* possui um modo de depuração que permite interagir facilmente com o *STM32*, agilizando esse processo de gravação dos programas. Para essa interação ocorrer com sucesso é necessário o uso de uma plataforma de gravação, o chamado *ST-LINK V2*.

Assim como descrito pela própria fabricante, a *ST Microelectronics*, o dispositivo *ST-LINK* é um depurador e um programador, compatível com microcontroladores da família *STM8* e *STM32*, que se conectam por meio de interface serial do tipo *JTAG*, bastante comum em aplicações de gravação de programas em sistemas embarcados. Além da compatibilidade direta com a família de microcontroladores, outra vantagem no uso dessas plataformas é a segurança, pois estes dispositivos são construídos de modo a isolar o microcontrolador da porta *USB* a que está conectado, garantindo que erros de implementação no conjunto não causem danos permanentes à estação do programador.

Ao visitar o *site* da empresa ou revendedores oficiais destes produtos, será encontrado um modelo deste depurador que possui aspecto bastante diferente dos comercializados em lojas de produtos eletrônicos, tanto no mercado nacional, quanto internacional. De todo modo, mesmo com aparências físicas distintas, ambos os dispositivos cumprem o mesmo papel, diferenciando apenas no custo de aquisição.

Para a gravação efetiva, basta acessar os pinos *PA13* e *PA14*, encontrados na parte inferior da placa, como sugerido pela Figura 19, e conectá-los diretamente aos pinos *SWDIO* e *SWCLK*, respectivamente, do gravador, além do pino de alimentação de 3,3 V e do pino de referência *GND*. Para garantir a segurança, é importante desconectar qualquer fonte de alimentação externa do circuito, deixando apenas a tensão do próprio gravador alimentar o microcontrolador enquanto a gravação acontece.

Ao conectar o gravador ao computador pessoal, por meio da entrada *USB*, o *driver* de instalação deve configurar automaticamente o dispositivo, sem qualquer interação extra. Para realizar a gravação, utilizando o *TRUE STUDIO*, basta carregar a aplicação desenvolvida, compilar o programa, e clicar no botão "*Debug*", na barra

de ferramentas, fazendo com que o programa troque de contexto e a depuração seja iniciada. A usabilidade é bastante simples, necessitando de apenas um pouco de prática para efetuar a maior parte das funções essenciais.

Além da gravação de memória, alguns detalhes são importantes na manipulação da plataforma *STM32F103C8TX*:

- Não se deve utilizar a conexão *micro-USB* para a alimentação do sistema quando estiver disponível uma fonte externa conectada ao pino de +5 V, já que ambos são conectados internamente, o que pode provocar um curto entre os dispositivos e causar danos elétricos;
- Embora disponíveis para implementação, a manipulação dos pinos *PB13*, *PB14* e *PB15* como portas de entrada e saída só é recomendada em casos críticos, pois, como visto na Figura 19, a capacidade de corrente desses pinos é reduzida e estão diretamente conectados aos osciladores internos responsáveis pelo *clock* global. Um dano elétrico nesses pinos pode resultar em prejuízos irreversíveis no microcontrolador todo.

4.4 Desenvolvimento de Aplicativo para Dispositivos Móveis

Ao citar estudos voltados à robótica, o foco do desenvolvimento é normalmente voltado totalmente para a parte embarcada do sistema, apenas a nível de *hardware* e *software*, sem desenvolver muitas discussões em relação a aplicativos que possam agregar no ecossistema da solução. Muitas vezes, o intuito principal do projeto nem é voltado a uma plataforma móvel, entretanto, esse tipo de ferramenta pode servir como apoio para o desenvolvimento, permitindo a disponibilização de dados relevantes de maneira mais legível, enquanto o programa passa por uma depuração.

O método mais comum utilizado para a criação destes aplicativos envolve o uso de ferramentas dedicadas, baseadas em linguagens de programação com suporte a elementos gráficos e interação com os periféricos comumente encontrados em dispositivos móveis. Entretanto, para suprir as necessidades didáticas discutidas até aqui, a construção do *kit* não deve depender diretamente de conhecimentos prévios, incentivando a busca por alternativas que possam cumprir este requisito, garantindo que não apenas o aplicativo seja simples, mas sua construção também apresente essa característica.

Como resultado dessas buscas, uma ferramenta que se encaixa adequadamente nos limites propostos, de fácil acesso e totalmente gratuita, surgiu como proposta para integração da aplicação proposta: a plataforma *online MIT App Inventor*. Anteriormente mantida pelo grupo *Google*, o *App Inventor* é uma ferramenta de código aberto que compreende um ambiente virtual de desenvolvimento, próprio para

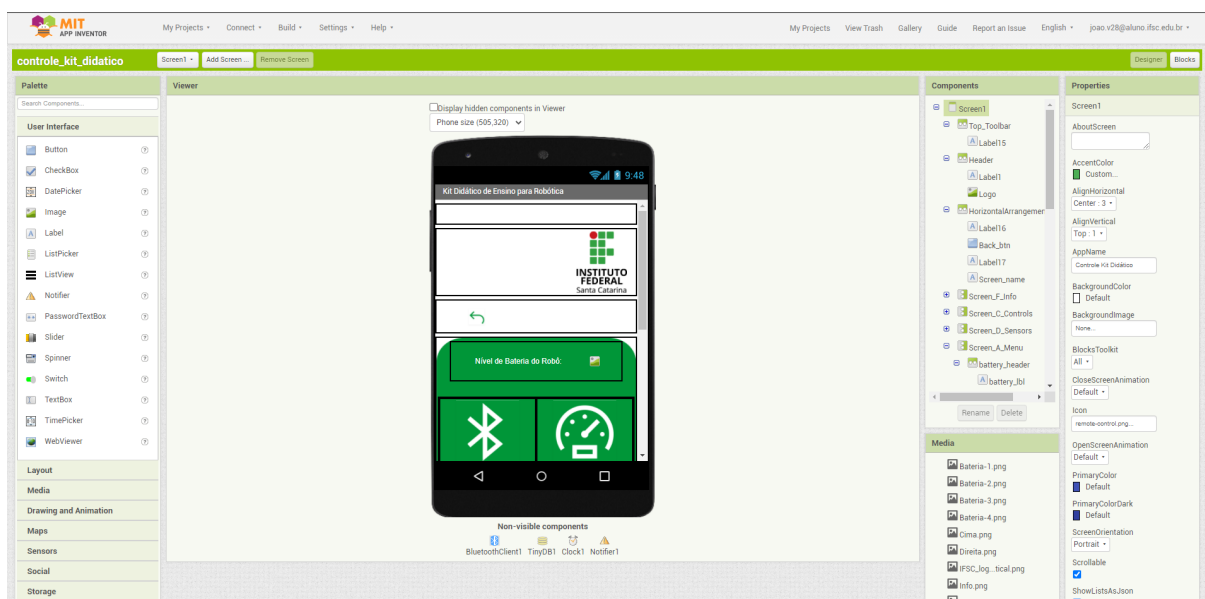
a construção de aplicativos simples destinados a dispositivos móveis, com suporte a programas que rodam no sistema operacional *Android*. A maior vantagem deste sistema é a acessibilidade, já que não exige conhecimento prévio do desenvolvedor, apresenta uma extensa documentação e tutoriais simples disponibilizados na própria plataforma e é programado por meio de ferramentas visuais intuitivas.

Para melhor entendimento do uso desta plataforma, os tópicos a seguir cobrem o desenvolvimento de um aplicativo que apresenta funcionalidades simples compatíveis com o robô desenvolvido neste projeto, focando principalmente no estabelecimento da lógica de interação dos sistemas.

4.4.1 Interface Gráfica de Usuário

Ao criar um novo projeto pelo *website* <appinventor.mit.edu/>, o primeiro contato do usuário com o ambiente de desenvolvimento será por meio da tela de *design* do aplicativo. Nessa tela, uma série de menus permite a seleção de elementos gráficos e funcionais, abrangendo a maior parte dos mecanismos de interação vistos em um aplicativo comum, como botões, caixas de seleção e itens de organização de *layout*. Para um melhor entendimento do aspecto visual, que baseia essa ferramenta de criação de aplicativos, a Figura 41 evidencia a página inicial do projeto, na qual se pode ver o menu de seleção de elementos (à esquerda da tela), a interface de integração gráfica (ao centro) e os componentes já selecionados, acompanhados das configurações particulares de cada um deles (à direita da página).

Figura 41 – Ambiente de edição para o *design* do aplicativo no MIT App Inventor



Fonte: Adaptado de MIT App Inventor.

O processo de inclusão de novos elementos é bastante simples, literalmente arrastando o tipo de componente desejado da barra esquerda para o centro da tela, o posicionando na região desejada dos limites que representam um dispositivo móvel. Depois dessa seleção, o elemento escolhido passa a fazer parte da aba Componentes e, ao ser selecionado, abre um menu que permite personalizar características como cores de fundo, tamanho de fonte, alinhamento e proporções de espaço na tela. Em cada um destes menus individuais também constam aspectos relacionados à funcionalidade do elemento, permitindo a edição destes parâmetros de modo visual.

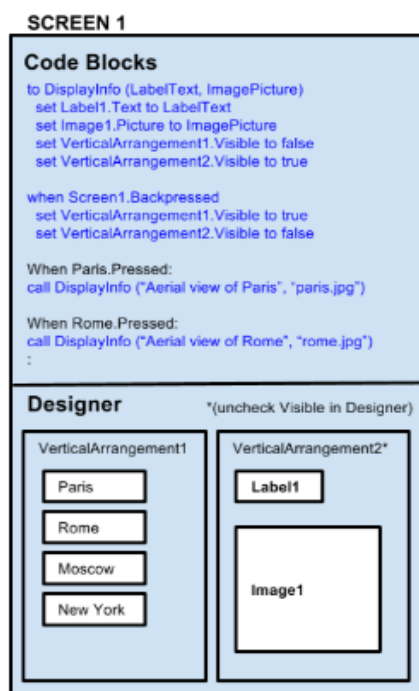
Uma das condições principais para a criação de um aplicativo, com uma melhor experiência de usuário, é a capacidade de lidar com múltiplas telas que, no *App Inventor*, é implementada por meio de duas maneiras principais de gerenciamento destes itens, previstas pela própria documentação da plataforma: a instância de telas distintas e o uso das chamadas telas "virtuais". A primeira opção, que pode ser considerada como "oficial", prevê a criação de telas como interfaces que trabalham individualmente, só interagem entre si nas trocas de contexto e apresentam elementos e organizações separados. O uso deste tipo de configuração é muito interessante, pois facilita a organização de *layout* dos componentes e posteriormente na programação dos blocos. É importante lembrar que a documentação do *App Inventor* recomenda um número máximo de 10 telas distintas, cuidado que visa evitar uma sobrecarga do processamento da maioria dos dispositivos baseados em sistemas *Android*.

Entretanto, uma limitação prática dessa plataforma, responsável por um grande impacto na construção dos aplicativos, é a sua impossibilidade de compartilhamento de algumas funcionalidades entre telas diferentes, impedindo que determinada função, que possua papel global na aplicação, possa ser acessada simultaneamente em múltiplas telas. O exemplo mais claro desse contexto, que se tornou a situação impeditiva do uso dessa configuração para a solução que compreende o *kit* didático, é a utilização do módulo de cliente da conexão *Bluetooth*, que pode ser associado a todas as telas existentes, mas que, ao fazer a troca de contexto entre telas, derruba a conexão e desconecta de todos os dispositivos vinculados. Essa impossibilidade proibiu a criação de uma tela de configuração da conexão por *Bluetooth* em uma janela separada de outras definições de função do aplicativo.

Para resolver este problema de maneira simples, é possível utilizar o segundo modo de gerenciamento da interface visual, criando as chamadas "telas virtuais", que utilizam apenas uma instância de tela (pensando nesta como um item distinto na plataforma) e dependem da manipulação dos parâmetros de visibilidade de cada elemento para criar uma lógica que esconde ou mostra os conjuntos, dependendo da visualização selecionada pelo usuário. Este tipo de implementação utiliza tanto características de *layout* proporcionadas pelo editor de *design*, quanto os comportamento

da programação encontrada no ambiente de desenvolvimento em blocos. A Figura 42 representa um exemplo de diagrama de blocos que utiliza como base o modelo de tela virtual.

Figura 42 – Estrutura de blocos que representa a construção de uma tela virtual



Fonte: (MIT, c. 2012-2015).

Além de exemplificar a criação de uma tela virtual, esse diagrama ilustra a criação de um aplicativo como um todo: uma tela virtual pode ser caracterizada por um *Vertical Arrangement*, que é normalmente utilizado para criar condições de alinhamento diferentes das definições gerais da tela completa, dotado de um parâmetro de visibilidade, passível de ser habilitado ou desabilitado, escondendo consigo todos os elementos que estiverem dentro de seu domínio, independentemente da visibilidade individual de cada um. Os elementos de *layout* de arranjos horizontais e verticais são essenciais para o aspecto visual do aplicativo, pois facilitam a distribuição e associação dos elementos individuais.

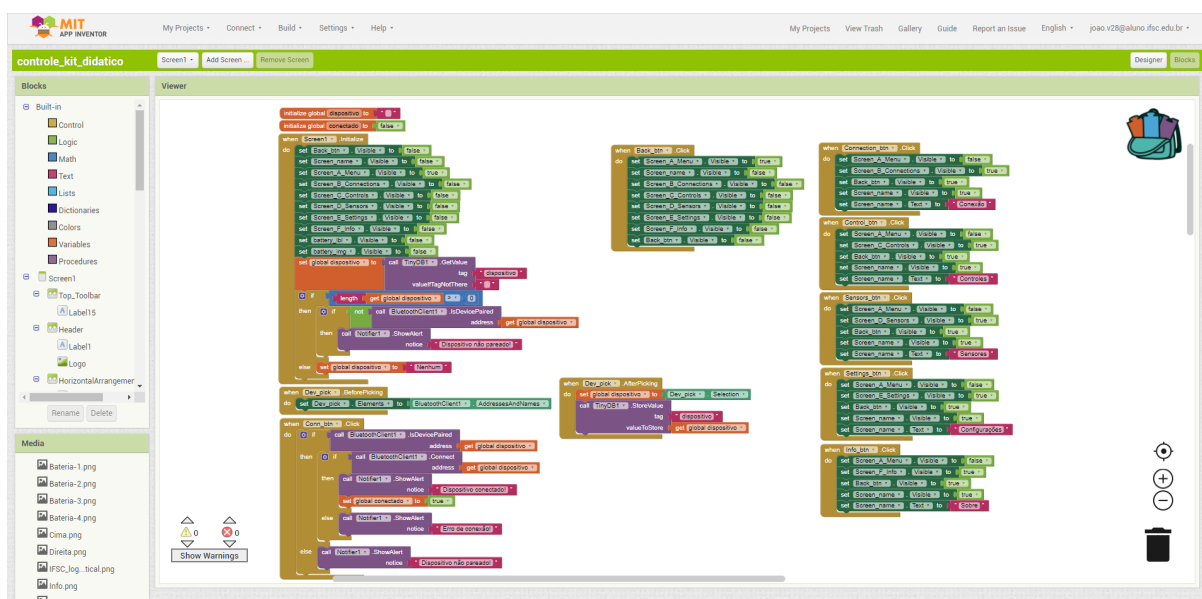
Descrevendo o funcionamento do exemplo proposto na Figura 42, uma tela virtual principal com quatro botões é colocada em um *VerticalArrangement1*, de visibilidade habilitada por padrão. Ao pressionar qualquer um dos botões, enumerados com nomes de cidades famosas, uma rotina é chamada e realiza o processo de modificar o texto e a imagem vinculada para as informações pertinentes ao botão e, em seguida, torna o *VerticalArrangement1* invisível, ao passo que o *VerticalArrangement2* aparece e exibe as informações da cidade escolhida. Para retornar, um clique em qualquer ponto do fundo da tela inverte as condições de visibilidade e o programa volta

ao estado inicial. Essa troca total de contexto é possível por conta da característica de um arranjo vertical controlar a visibilidade de todos os elementos de seu domínio, funcionando como "telas".

4.4.2 Ambiente de Programação em Blocos

Outro ponto forte da ferramenta é seu ambiente de programação, responsável por definir os comportamentos individuais ou de interação para cada um dos elementos presentes no *layout*, programando por meio de uma interface de blocos as reações a cada decisão e interferência do usuário, podendo ser um botão que leva a outra tela, um seletor de dispositivos ou um *slider* para definição de volume. Para uma visualização dessa modalidade de programação em blocos apresentada pela plataforma, a Figura 43 evidencia a janela que é disponibilizada ao usuário quando ele faz a transição da aba *Designer* para a sua alternativa, a aba 'Blocos'.

Figura 43 – Ambiente de programação em blocos do MIT App Inventor



Fonte: Adaptado de MIT App Inventor.

Para estabelecer as definições de comportamento dos elementos da tela, são imaginadas estruturas isoladas individuais a cada ação dos componentes, como se todas elas fossem espaços definidos do programa que executam códigos sequenciais de maneira concorrente, verificado constantemente pelo processamento do dispositivo, acionando cada bloco conforme a necessidade funcional. Por exemplo, um bloco pode ser criado para executar uma rotina específica após pressionar um botão, outro pode apresentar uma reação à abertura de um seletor de opções e um terceiro, após a escolha nesse mesmo seletor, processar as configurações escolhidas. Cada setor

desses pode ser trazido para a interface de programação por meio da barra lateral esquerda, a qual apresenta os componentes selecionados na tela de *design* e, ao clicar em cada um, explicita quais as ações podem ser personalizadas para este elemento.

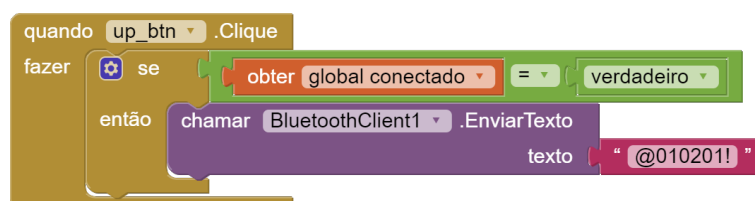
4.4.3 Organização dos eventos de interface

Uma grande parcela do projeto de um aplicativo, além da parte de construção técnica das funcionalidades, é o processo de apresentar ao usuário da plataforma uma maneira dinâmica e intuitiva de interagir com as peculiaridades da aplicação, garantindo uma experiência mais amigável e consistente. Nesse sentido, entender como os eventos de sistema são gerados é de grande importância, pois reflete no fluxo de informações, internas ao processamento ou explicitamente visíveis a quem utiliza, requerendo diferentes elementos de interação.

No aplicativo desenvolvido para este projeto, por seu conceito simplificado baseado na troca de algumas informações entre o robô e um dispositivo móvel, os eventos de sistema tem basicamente duas fontes distintas: os botões de ação que disparam comandos na direção aplicativo-robô e a recepção de mensagens via *bluetooth* que gera eventos na direção robô-aplicativo. Ambas as naturezas de informação valem-se das estruturas básicas de ação discutidas anteriormente para processar pacotes de dados estabelecidos em padrão inteligível ao sistema.

Em meio as diferentes telas disponíveis no aplicativo, uma série de botões (além dos que são definidos para facilitar a navegação entre os menus) é utilizada para gerar diferentes tipos de comandos, que são montados em forma de *strings*, respeitando as estruturas propostas na subseção 4.3.7 para o protocolo de comunicação entre os sistemas, e enviados via *bluetooth* para estimular diferentes respostas no conjunto embarcado. Um exemplo dessa configuração são os botões utilizados para controlar o movimento do protótipo, que é feita por meio de quatro botões direcionais e um extra, capaz de cessar o movimento dos motores, cada um deles vinculado a uma ação que envia um pacote específico de informação contendo o código de mensagem vinculado ao controle e um parâmetro adicional que indica a direção desejada. Na Figura 44, exemplifica-se o uso de um bloco de ação que envia o comando necessário para que o carrinho realize um deslocamento na direção frontal.

Figura 44 – Bloco de execução de evento de botão

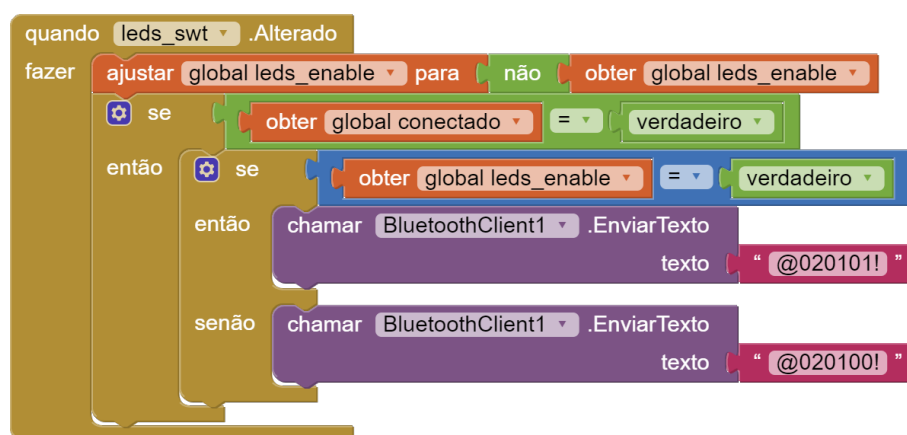


Fonte: Adaptado de MIT App Inventor.

Nomeado *up_btn*, ao ser pressionado, o botão reflete em uma ação simples que envia uma sequência de símbolos, começando com o caractere especial "@", seguido do código de mensagem "01"(primeiro *byte*), que sinaliza um comando de direção de movimento. Depois, a mensagem segue com um primeiro parâmetro "02"(segundo *byte*), que indica a seleção de uma direção específica, e um último parâmetro "01"(terceiro *byte*), que se refere à direção frontal de movimento. Por fim, para fechar o pacote aos moldes do protocolo, é adicionado um caractere especial "!" de finalização, que garante um nível simples de verificação da integridade do pacote recebido. É interessante notar que, por lidar com as funções de comunicação *bluetooth*, antes de enviar a mensagem propriamente dita, é preciso verificar se existe algum dispositivo conectado por meio da variável global "*conectado*".

Outro exemplo conveniente para analisar essa construção dos eventos do sistema é o reflexo da variação de estado do elemento *switch*, responsável por habilitar ou desabilitar determinadas funcionalidades, que também possui o papel de construir um pacote de mensagem adequado ao estado atingido na ação de transição de estados. Na Figura 45, está explicitado o bloco de ação que executa a troca de contexto para iniciar ou parar a rotina de iluminação dos *LEDs*.

Figura 45 – Controle para habilitar ou desabilitar funcionalidade do sistema

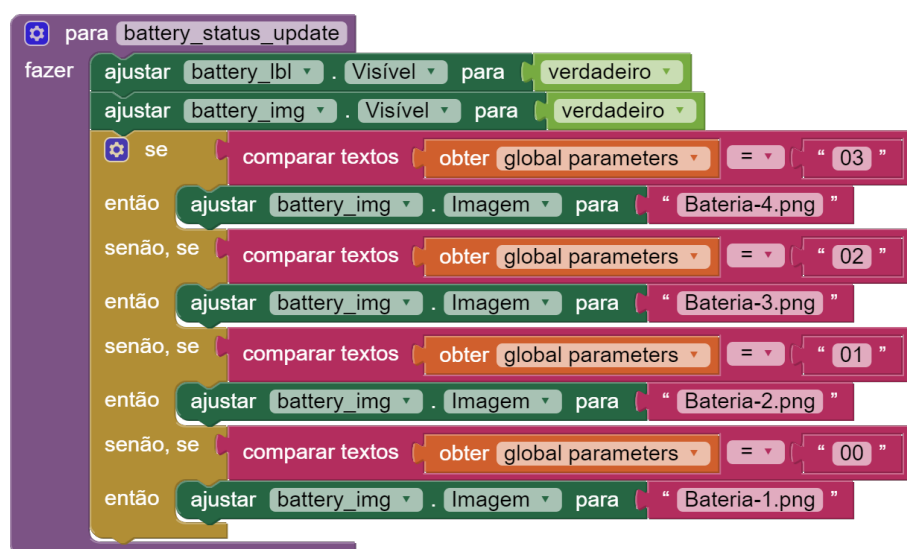


Fonte: Adaptado de MIT App Inventor.

Com identificação *leds_swt*, ao ter seu estado modificado, a chave de seleção apenas anuncia que existiu uma troca de contexto, deixando a cargo de uma variável auxiliar a persistência do estado de configuração, sendo esta atualizada toda vez que o bloco de transição de estado for chamado para executar sua rotina. Após atualizar o estado, nos mesmos moldes do botão de direção, a rotina envia um pacote referente ao novo estado atribuído, permitindo ao robô identificar qual configuração deve ser estabelecida no contexto de execução. Novamente, os critérios de verificação da correta utilização das funções do cliente *bluetooth* são avaliados, permitindo um melhor gerenciamento das conexões.

Falando sobre a outra direção de transmissão de dados, vindos do robô em direção ao aplicativo, o processo de identificação das informações é muito parecido, apenas com o diferencial de passar por uma função de decomposição do pacote recebido e posteriormente ser interpretada pela rotina que identifica o código da mensagem, chamando a sequência de programa adequada para tratar os dados recebidos. Um exemplo desse processo é o tratamento feito para anunciar ao usuário o nível de carga atual da bateria conectada ao robô, que depende de mensagens enviadas constantemente pelo próprio sistema embarcado e atualiza um identificador visual diferente conforme o nível lido. Na Figura 46, está descrita a sequência de código que seleciona a imagem adequada do nível de bateria de acordo com o parâmetro recebido via comando *bluetooth*.

Figura 46 – Acompanhamento remoto do *status* da bateria do robô



Fonte: Adaptado de MIT App Inventor.

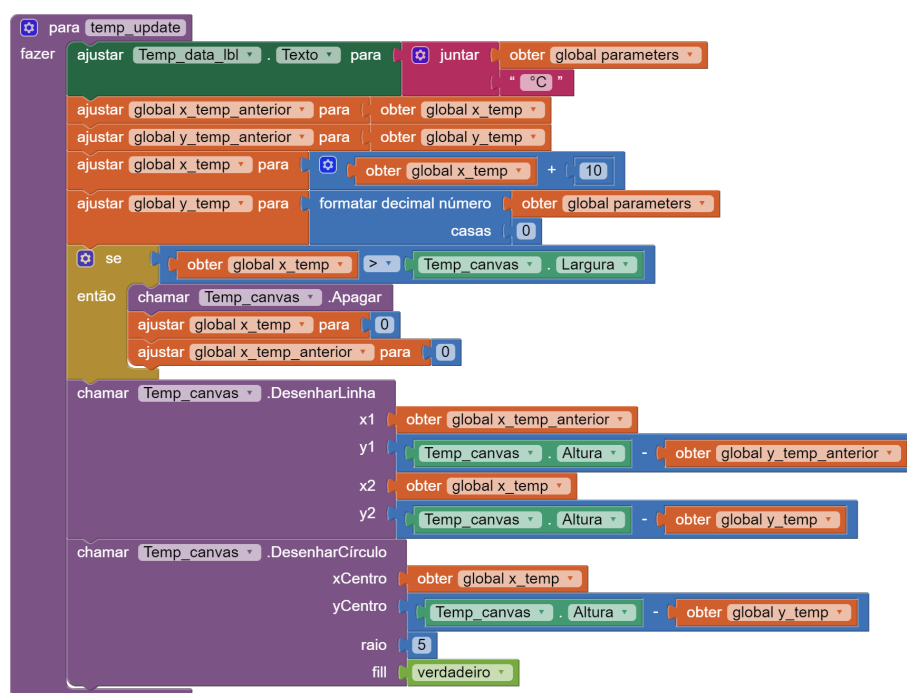
Tanto o texto quanto a imagem relacionados ao nível de bateria estão presentes o tempo todo na primeira tela, entretanto, só tem sua visibilidade ativada no momento em que a primeira mensagem de atualização de *status* for recebida, o que é atualizado pela rotina *battery_status_update*. Depois de habilitar o elemento visual, a sequência de comando de condição identifica qual o código do estado lido e atribui uma das imagens definidas previamente para evidenciar o estado atual da carga da bateria.

Outro processamento que pode ser realizado com os dados recebidos é um acompanhamento progressivo da evolução dos dados de entrada, principalmente para as informações que possuem valores numéricos e significados práticos para o usuário. Como o sistema apresenta diferentes sensores para a aquisição de sinais do ambiente, acompanhar periodicamente a mudança dos valores lidos é bastante interessante ao

contexto, podendo refletir em tabelas ou até mesmo gráficos de linha. A plataforma *App Inventor* possui suporte para elementos visuais e permite a manipulação de imagens para a construção de formas simples construídas com referência em dados de entrada, criando desenhos por meio de manipulações matemáticas simples, facilmente implementáveis.

Para a aplicação aqui desenvolvida, três variáveis de interesse tiveram seus dados refletidos em gráficos que, ao receber um novo valor, cria uma linha reta entre o último ponto identificado e o novo ponto, permitindo esse acompanhamento desejado para o histórico de um determinado número de amostras. Na Figura 47, está exemplificada a rotina que lida com a temperatura lida pelo módulo do sistema embarcado e recebida nas mensagens, sequência que é repetida e adaptada para as outras variáveis de interesse presentes no aplicativo. Além de mostrar em formato de texto o último valor recebido, a rotina identifica o ponto anterior para o mesmo sinal e cria uma linha entre este e o novo dado de entrada, marcando também um círculo que representa graficamente esse novo valor lido.

Figura 47 – Criação de um gráfico com a variável de interesse



Fonte: Adaptado de MIT App Inventor.

Para executar as ações relacionadas à criação de gráficos, o *App Inventor* faz uso de um elemento com a alcunha de *canvas*, que habilita diferentes funcionalidades de construção para formas geométricas com base em conceitos simples de geometria analítica. No exemplo verificado, duas dessa ferramentas são utilizadas, uma para simbolizar a evolução do valor anterior para o atual, por meio de uma linha reta,

enquanto a outra desenha um pequeno círculo, que marca a intensidade da amostra recebida normalizada ao eixo vertical. Além disso, outra função relacionada ao *canvas* faz a limpeza deste no momento que todo o espaço destinado a suas informações for preenchido, permitindo que o processo reinicie e novas informações possam ser processadas.

4.4.4 Descrição Geral das Funcionalidades do Aplicativo Móvel

A versão final do aplicativo foi desenvolvida por meio da utilização de todas as funcionalidades descritas até aqui, definindo comportamentos básicos de interação com o robô, incluindo o envio de comandos e a atualização periódica das informações lidas pelos sensores. A dinâmica de navegação entre as funcionalidades disponíveis foi pensada de modo a fazer uso do conceito de telas virtuais mencionado anteriormente, facilitando a organização das diferentes informações disponibilizadas para os usuários. De maneira a explicitar o contexto básico que envolve a utilização do aplicativo móvel proposto, nos próximos parágrafos serão apresentadas as telas criadas e as opções de interação disponível em cada uma atribuída a elas, facilitando a utilização desta ferramenta.

Figura 48 – Tela de início do aplicativo móvel



Fonte: Autoria Própria.

Ao abrir a aplicação, o usuário é apresentado a uma tela inicial simples que representa o menu principal com todas as páginas secundárias disponíveis para acesso, cinco no total, que permitem atuar sobre o robô ou mesmo verificar seu estado de funcionamento. Este menu é composto por cinco botões distintos, com imagens próprias e descrições que sugerem suas funcionalidades, com distribuição que segue os padrões vistos na Figura 48, na qual é possível verificar a captura de tela de um dispositivo móvel, com sistema operacional *Android*, rodando a aplicação desenvolvida e exibindo essa tela de início.

Como mencionado anteriormente, uma informação relevante, que também é apresentada na tela inicial, diz respeito ao estado de carga da bateria, lido pelo robô e atualizado periodicamente por meio de mensagens recebidas pelo cliente *bluetooth*, entretanto, apenas habilitado quando a primeira mensagem com código "0x04" (*status* da bateria) for identificada. Enquanto esse pacote não for recebido pelo aplicativo, nenhuma informação é mostrada e a condição de visibilidade para ambos os elementos (texto e imagem) é mantida desabilitada. Quando recebe o primeiro pacote de verificação desse, o sistema seleciona uma entre as quatro imagens, com níveis diferentes de carga, permitindo ao usuário identificar se a bateria está funcionando em seu estado crítico e uma recarga deve ser executada.

Ao clicar no primeiro botão disponível na tela inicial, marcado com a identificação "Conexão", o programa realiza a troca de contexto envolvendo as telas virtuais e esconde os elementos que fazem parte do menu principal, passando a habilitar uma tela que tem por objetivo permitir ao usuário realizar o processo de conexão do aplicativo com o módulo *bluetooth* do carrinho. Sobre as funcionalidades, o menu de conexão apresenta apenas dois botões distintos, que são responsáveis por completar o processo necessário de seleção do dispositivo de destino e efetivamente interligar os sistemas. Na Figura 49, uma captura de tela evidencia o aspecto visual deste menu.

Para deixar mais intuitivo o processo de configuração, que é necessário para a utilização do aplicativo como controle, um pequeno tutorial contendo uma sequência básica de passos para executar traz ao usuário uma maneira fácil de consultar e completar a interação inicial. Por meio do sistema de notificações que é vinculado à aplicação, o usuário ficará ciente se ocorrer algum problema de conexão, o obrigando a voltar para este menu e a realizar novamente a sequência de passos descrita.

Além disso, como ferramenta de navegação, ao entrar em um dos menus secundários, além do título da página, surge também um outro botão marcado com a imagem de uma seta de retorno, que permite voltar à tela principal e acessar novamente este e os outros menus secundários, desaparecendo assim que adentrar novamente na tela de início.

Figura 49 – Tela para configuração da conexão *Bluetooth*

Fonte: Autoria Própria.

Por meio destes quatro botões direcionais e de um outro, marcado por um círculo, que obriga o sistema a parar, o usuário pode manipular a direção do carrinho de forma intuitiva e simplificada, enviando automaticamente os comandos equivalentes às mudanças de direção, previstos em protocolo. Outra interação que pode resultar em comandos para o carrinho é um seletor de duas posições, que foi configurado para habilitar ou desabilitar o funcionamento do sistema em seu modo seguidor de linha, passando a fazer uso dos sensores ópticos *TCRT5000* para se posicionar geograficamente. Uma informação adicionada à tela é de que o modo seguidor de linha desabilita a possibilidade de interação por meio das setas direcionais, entretanto, essa decisão é parte da rotina de *software* e a implementação deste comportamento foi realizada no programa do próprio sistema embarcado.

A segunda tela, iniciada pelo botão marcado como "Controles, acessível pelo menu principal, apresenta os controles direcionais que podem interferir na movimentação do carrinho. Na Figura 50 é apresentado esse menu secundário por meio de uma captura da tela do dispositivo móvel que roda a aplicação.

Figura 50 – Tela com os controles de direção do robô

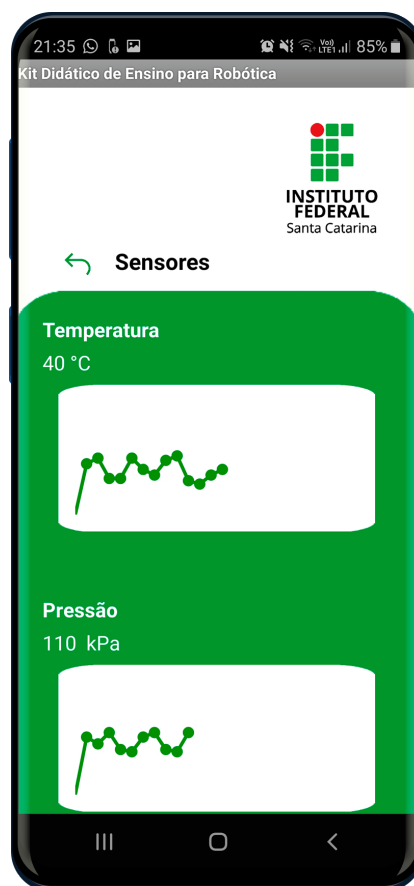


Fonte: Autoria Própria.

Para a interação com os sensores presentes no *kit*, uma terceira tela foi adicionada e construída pensando em maneiras intuitivas de apresentar a informação ao usuário, garantindo legibilidade e significado às informações recebidas. Na Figura 51, pode ser vista a captura da tela de um dispositivo móvel que roda o aplicativo e mostra a tela de acompanhamento do estado dos sensores.

Fazendo uso da capacidade da plataforma *App Inventor* de adicionar funcionalidades visuais de construção geométrica à aplicação, além de mostrar o último valor, cada variável de ambiente acompanhada pelo robô traz consigo um gráfico evidenciando as últimas amostras recebidas e permitindo visualizar uma progressão desses valores. Embora no trecho mostrado apenas duas variáveis diferentes possam ser identificadas, a tela possui rolagem vertical e uma terceira informação, relacionada a medidas de altitude, pode ser acessada e repete a maneira de expressar os dados vistos nas duas primeiras instâncias. A atualização dos gráficos é feita individualmente, a cada vez que um evento de atualização de algum dos sensores for disparado.

Figura 51 – Tela de acompanhamento dos estados dos sensores



Fonte: Autoria Própria.

O próximo menu que pode ser acessado na tela inicial, por meio do botão de "Configuração", é a tela que permite realizar algumas configurações básicas voltadas à manipulação dos elementos de sinalização de eventos conectados ao sistema, neste caso, permitindo o controle de comportamento sobre os *LEDs* e o *buzzer*. Na Figura 52, é possível identificar as diferentes interações permitidas pela aplicação.

A tela é responsável por apresentar a possibilidade de habilitar e desabilitar as funcionalidades individuais de funcionamento das duas categorias de sinalização, além de garantir alterações de volume do *buzzer* e escolher algumas rotinas demonstrativas de mudanças de estado dos *LEDs*. Após a seleção de cada uma das informações, um comando é gerado nos parâmetros do protocolo de comunicação *bluetooth* e envia ao robô as decisões do usuário.

Figura 52 – Tela com configurações do sistema



Fonte: Autoria Própria.

Um último botão adicionado à tela principal tem o intuito de dar acesso a uma parte burocrática do aplicativo, que apresenta informações de versão e da motivação de criação e uso do mesmo, além de alguns alertas de direitos autorais da plataforma utilizada e das imagens selecionadas para ilustrar os elementos visuais. Na Figura 53, a captura de tela do dispositivo móvel que roda o aplicativo evidencia algumas das informações disponíveis, servindo de caráter informativo.

Embora algumas interações de usuários apresentadas nesta solução não sejam essenciais ao controle do carrinho, as funcionalidades de interface escolhidas para fazer parte deste aplicativo foram adicionadas com a ideia de provar conceitos e exemplificar o uso de diferentes elementos da plataforma *App Inventor*, permitindo imaginar as possibilidades de personalização e as funções que podem fazer parte de uma implementação futura. Entretanto, mesmo apresentando uma parte destes elementos, grande parte das funcionalidades disponíveis não foram utilizadas, o que é maximizado pelo fato de que a plataforma aceita a importação de extensões desenvolvidas por terceiros, facilmente encontradas em pesquisas na *web* e responsáveis pelas mais diversas melhorias de capacidade de implementação de diferentes programas.

Figura 53 – Tela com informações sobre o aplicativo



Fonte: Autoria Própria.

5 RESULTADOS OBTIDOS

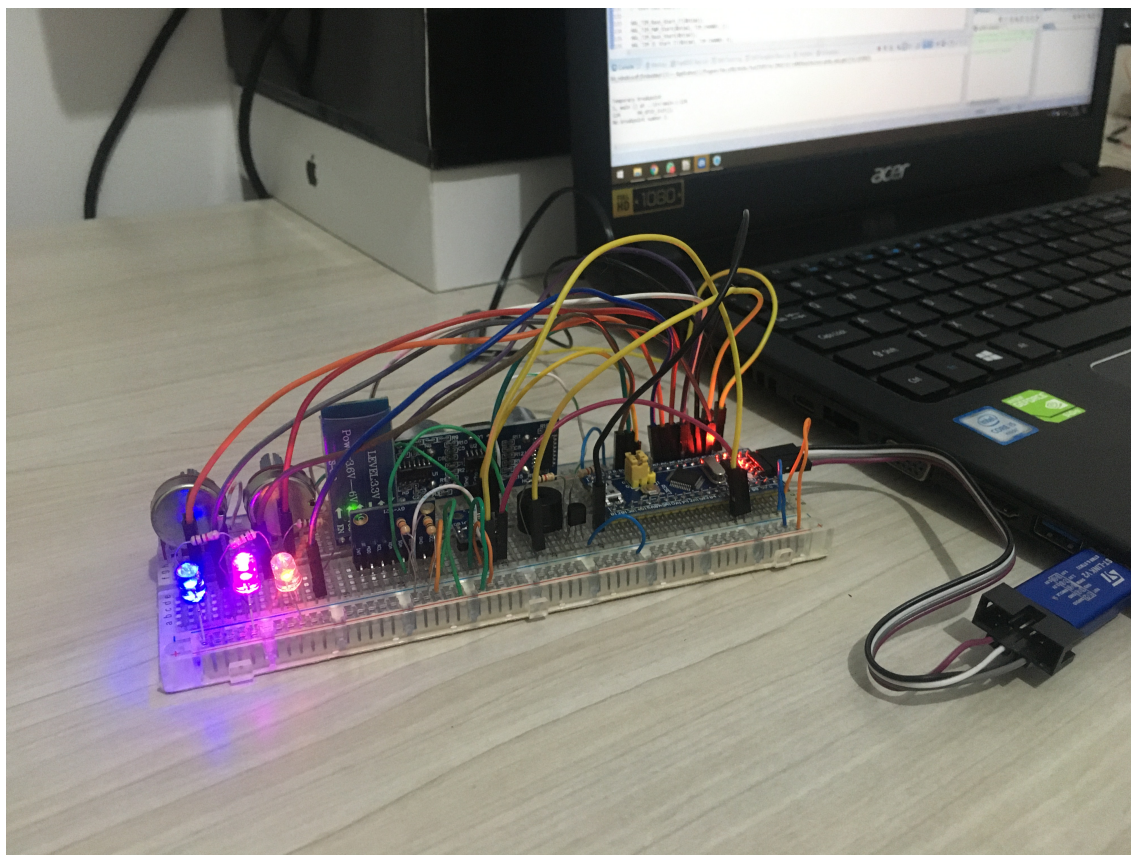
Por ter a intenção de concorrer com alguns conjuntos de referência, este projeto deve ter suas características principais comparadas com as alternativas apresentadas anteriormente, verificando se as funcionalidades desenvolvidas atingiram os critérios propostos para suas implementações. Os tópicos a seguir tratam da validação do processo de confecção do *kit* didático proposto, fazendo uma análise de custo e funcionalidade, além de discutir os resultados obtidos em um contexto geral.

5.1 Testes Finais

Embora o desenvolvimento deste projeto tenha se iniciado no primeiro semestre de 2020, com conclusão prevista para o mês de julho, ainda em março de 2020, no início das atividades práticas, se desencadeou a pandemia de COVID-19 que requereu medidas de isolamento sérias, impedindo o uso dos laboratórios de desenvolvimento do IFSC e limitando os recursos para a confecção do robô, de acordo com as especificações definidas previamente. Grande parte da pesquisa teórica e do desenvolvimento inicial pôde ser realizado de maneira remota, sem o auxílio das ferramentas de bancada presentes nos laboratórios, porém, como a construção de uma placa de circuito impresso depende de insumos e processos específicos, a montagem do protótipo final sofreu uma grande baixa e não pôde ser executada.

O único procedimento que teve uma execução prática possibilitada foi a montagem dos subconjuntos em matriz de contato, que permitiu a validação prática das conexões necessárias e alguns ajustes isolados de circuito, principalmente relacionados ao uso dos módulos de sensoramento e suas interfaces de comunicação. De maneira geral, todos os circuitos que possuíam baixo perfil de consumo e que tinham seu funcionamento diretamente atrelados ao microcontrolador, foram montados para a realização das provas de conceito, garantindo que as configurações feitas em *software* refletiriam nos comportamentos adequados. Na Figura 54, está apresentada uma imagem capturada da matriz de contatos em funcionamento, montada com os sensores e elementos de notificação visual e sonora, além da emulação da leitura do *ADC* e da possibilidade de comunicação via *bluetooth* com dispositivos externos.

Outra parte do sistema que foi possível validar foi o aplicativo móvel desenvolvido e sua interação com a placa, utilizando inicialmente apenas rotinas de teste simples para a verificação da troca correta de dados, progredindo para o uso destas informações como comandos que interferem nos comportamentos dos dois lados da conexão e, por fim, um refinamento de usabilidade, tanto na aplicação visual quanto no sistema embarcado, tornando o aplicativo mais funcional. Para garantir o comportamento correto em diferentes plataformas, o aplicativo móvel foi testado em

Figura 54 – Montagem do sistema em matriz de contatos

Fonte: (Autoria Própria, 2020).

três dispositivos diferentes, todos gerenciados pelo sistema operacional *Android*, sem apresentar qualquer erro que inviabilizasse o uso deste na interação com o projeto.

Analisando estes subconjuntos vinculados ao sistema, talvez o que mais dependa de testes de bancada para ajustes seja o circuito de acionamento dos motores que, por apresentar uma potência maior, não pôde ser testado sem uma alimentação auxiliada por fontes externas ou mesmo por baterias, que não foram adquiridas a tempo e dependeriam de um condicionamento que seria proporcionado pela construção da placa. Desse modo, para garantir que essa parcela crucial do sistema esteja em plena operação, é necessário que seja feita a montagem da placa em seu estado final, ficando como sugestão para intervenções futuras.

5.2 Análise de custos

Para ter um entendimento melhor dos resultados atingidos, pela existência de uma preocupação com o aspecto econômico envolvido na produção do *kit* didático descrito aqui, é importante desenvolver uma verificação dos custos envolvidos no processo, principalmente relacionados aos insumos básicos que compõem a estru-

tura principal. No intuito de proporcionar uma noção prática dos gastos necessários, a estimativa de custos dos principais componentes empregados é apresentada na Tabela 9.

Avaliando individualmente os preços para cada um dos componentes, é perceptível um custo final maior causado por algumas categorias específicas. Exemplos destes componentes são os módulos de sensoriamento, por se caracterizarem como soluções completas. Um exemplo factível, que pode servir como comparação neste caso, é o do sensor óptico *TCRT5000*: em uma consulta ao website <www.eletrogate.com> é possível encontrar um módulo pronto para leitura de um sensor óptico por R\$ 8,90, totalizando R\$ 17,80 para o uso de 2 módulos na execução da rotina de seguidor de linha; em comparação, tomando o circuito utilizado neste projeto e estimando o custo somado dos componentes utilizados (2 sensores *TCRT5000*, 4 resistores, 2 potenciômetros e um CI amplificador operacional), para empregar os dois pontos de leitura necessários, o custo total é de aproximadamente R\$ 8,69, menos da metade do preço.

Claramente, nem todos os módulos têm sua complexidade reduzida a ponto de serem facilmente substituídos por um circuito simples, principalmente se tratando de *kits* didáticos para usuários com menor familiaridade com eletrônica. Deste modo, para a escolha adequada do caminho utilizado na seleção dos sensores ao circuito, é interessante avaliar o custo-benefício, procurando um equilíbrio entre o preço de construção e a complexidade de implementação, inerente à interface de interação utilizada pelo mesmo.

Além dos sensores, um alto custo envolve a aquisição das baterias e da parte mecânica do sistema, principalmente dos motores, que são responsáveis pela movimentação autônoma do robô. Neste caso, os motores foram adquiridos juntamente com o chassi e as rodas compatíveis com eles, em um conjunto que normalmente é comercializado para a construção de plataformas como esta. Por meio de pesquisas rápidas, foi visto que a compra dos motores individualmente, contando apenas com as adaptações do eixo deste, levaria o custo a valores muito próximos ou até mesmo superiores ao valor encontrado para o *kit* completo, que é acompanhado pela base de construção do carrinho, tornando esta a maneira mais viável de adquirir os motores para este robô.

Para cumprir este mesmo objetivo, é incentivada também a busca por alternativas que reutilizam peças de eletrônicos fora de operação, podendo até mesmo empregar brinquedos inutilizados que em algum momento serviram para o mesmo propósito, como carrinhos de controle remoto, que apresentam a base mecânica adequada e motores compatíveis acoplados a ela, necessitando apenas da adaptação de alguns pontos do circuito e redimensionamento dos componentes envolvidos no acionamento.

Tabela 9 – Tabela de custos do projeto

Item	Qtde.	Unitário (R\$)	Total (R\$)
Resistor 1/4W 150 Ω /5% - PTH	4	0,08 (I)	0,32
Resistor 1/4W 390 Ω /5% - PTH	2	0,08 (I)	0,16
Resistor 1/4W 470 Ω /5% - PTH	9	0,08 (I)	0,72
Resistor 1/4W 1 k Ω /5% - PTH	3	0,08 (I)	0,24
Resistor 1/4W 2,2 k Ω /5% - PTH	1	0,08 (I)	0,08
Resistor 1/4W 10 k Ω /5% - PTH	7	0,08 (I)	0,56
Resistor 1/4W 15 k Ω /5% - PTH	2	0,08 (I)	0,16
Resistor 1/4W 30 k Ω /5% - PTH	2	0,08 (I)	0,16
Capacitor Multicamada 100 nF/50 V - PTH	2	0,38 (I)	0,76
Capacitor Eletrolítico 1 μ F/100 V - PTH	1	0,16 (I)	0,16
Potenciômetro 10 k Ω /16mm - Eixo Estriado	2	1,60 (I)	3,20
Diodo 1N4007	8	0,09 (I)	0,72
LED 5 mm	5	0,17 (I)	0,85
Transistor TJB BD139	4	0,97 (I)	3,88
Transistor TJB BD140	4	0,98 (I)	3,92
Transistor TJB BC548	5	0,20 (I)	1,00
CI Amplificador Operacional LM324	1	1,37 (I)	1,37
CI Comparador LM393	1	0,79 (I)	0,79
Microcontrolador BluePill STM32F103C6T8	1	32,90 (II)	32,90
Módulo de Comunicação Bluetooth HC-05	1	27,90 (III)	27,90
Módulo Sensor Ultrassônico HC-SR04	1	12,90 (IV)	12,90
Módulo Sensor de Temperatura e Pressão BMP180	1	16,90 (IV)	16,90
Módulo Sensor Acelerômetro de Giroscópio GY-521	1	18,90 (IV)	18,90
Sensor Óptico de Reflexão TCRT5000	2	1,90 (IV)	3,80
Buzzer passivo 5 V	1	3,50 (IV)	3,50
Chave Slide Switch	1	0,33 (V)	0,33
Chave Botão Tátil 6 x 6 x 4,3 mm	1	0,21 (I)	0,21
Regulador de Tensão Buck Step Down CC-CC MP1584	1	8,90 (IV)	8,90
Bateria LiPo 3,7 V/1,800 mAh	2	37,03 (III)	74,06
Conector Barra de Pinos Vertical Fêmea 1x02	4	0,46 (I)	1,84
Conector Barra de Pinos Vertical Fêmea 1x04	3	0,56 (I)	1,68
Conector Barra de Pinos Vertical Fêmea 1x08	2	0,67 (I)	1,34
Conector Barra de Pinos Vertical Fêmea 1x20	4	2,56 (I)	10,24
Chassi + 4 Motores + 4 Rodas (conjunto)	1	89,90 (IV)	89,90
Componentes	89	Total (R\$)	324,35

Fonte: Autoria Própria.

- (I) Valores de venda encontrados em <www.proesi.com.br> em Outubro de 2020
 (II) Valores de venda encontrados em <www.filipeflop.com.br> em Outubro de 2020
 (III) Valores de venda encontrados em <www.mercadolivre.com.br> em Outubro de 2020
 (IV) Valores de venda encontrados em <www.eletragate.com.br> em Abril de 2020
 (V) Valores de venda encontrados em <www.baudaeletronica.com.br> em Outubro de 2020

Durante a estimativa de preços foi levada em conta a acessibilidade dos mesmos, priorizando a compra de insumos em mercado nacional, que apresenta maior segurança na compra e um menor tempo de entrega. Não é desencorajada a compra de componentes por meio de comércio internacional, contudo, é alertado o cuidado em relação às tarifas de importação agregadas e outros critérios de comercialização impostos pelos revendedores. Outro fator que influencia diretamente na compra de produtos importados é a dificuldade de transporte até o consumidor que, além de demorada, comumente apresenta valores bastante elevados.

Como mencionado nas notas da tabela, os preços dos componentes comprados na loja online <www.eletrogate.com> foram verificados alguns meses antes dos demais, ocorrendo dessa forma pois, no início do projeto, foram adquiridos alguns componentes para a parte inicial do desenvolvimento, utilizando como insumos auxiliares resistores, capacitores e outros elementos básicos da eletrônica reutilizados de outros projetos, não os adquirindo especificamente para a execução deste. De todo modo, os valores descritos na tabela refletem os custos práticos da implementação do circuito proposto, podendo variar conforme o fornecedor e ser encontrado em condições superiores ou inferiores por meio de pesquisas de preço.

Ainda, embora facilitem o processo de conexão com os pontos elétricos da placa, o uso das barras de pinos no projeto são opcionais, reduzindo levemente o custo envolvido no projeto. O uso destes elementos auxiliares é indicado para manter a integridade no processo de soldagem para os subcircuitos conectados ao *kit*, como os módulos de sensores, ou mesmo facilitar o processo de substituição caso alguns destes apresente falhas críticas em seu funcionamento.

Por fim, é importante lembrar que não foram incluídos na análise individual de custos do conjunto alguns valores extras, como os que envolvem a confecção da placa de circuito impresso, a aquisição dos gravadores utilizados para escrever o programa no microcontrolador e a compra do carregador balanceador para as baterias *Li-Po*. Estes custos podem ser estimados separadamente, pois nenhum destes itens precisa ser exclusivo para cada placa confeccionada, diluindo o investimento financeiro realizado sobre eles. Na Tabela 10 é possível verificar o resultado da consulta de alguns preços relacionados a estes elementos.

Tabela 10 – Estimativa de custos extras para o protótipo

Item	Preço (US\$)	Preço (R\$)
PCI de confecção industrial (5 peças)	34,00	191,08*
Carregador de baterias balanceador	-	160,00 - 204,90
Gravador do modelo <i>ST-LINK V2</i>	-	29,90 - 48,90

Fonte: Autoria Própria.

(*) Conversão direta do valor em dólares, sob cotação de R\$ 5,62

Muitas plataformas de venda *online* são especializadas na comercialização de placas de circuito impresso, construídas por meio de processos industriais, aceitando, inclusive, pedidos em pequena escala. Algumas empresas brasileiras fazem parte deste ramo, entretanto, muitos projetos de PCI brasileiros ainda são fabricados com o auxílio de empresas internacionais, principalmente do mercado chinês, que apresenta boas condições para o custo de fabricação. Um exemplo de loja virtual dessa categoria é a *PCBWay* (<www.pcbway.com>), com a possibilidade de confecção destes projetos a um custo bastante reduzido.

Fazendo um orçamento de algumas peças por meio desta plataforma, para 5 placas com dimensões 145,4 x 74,5 mm, 1 camada e densidade de cobre igual a 1 onça por pé quadrado, o custo estimado de fabricação ficou em US\$ 24,00. Em acréscimo, o custo de transporte da placa até o Brasil, considerando a opção mais barata, com estimativa de entrega entre 82 e 97 dias, consistiu em US\$ 10,00, resultando em um custo total de US\$ 34,00. Como critério de comparação, a opção mais rápida de entrega, com prazo entre 5 e 7 dias, seria possível a um custo de US\$ 41,00. Fazendo a conversão direta do montante final, a uma cotação de R\$ 5,62 para o dólar americano, o valor gasto atinge os R\$ 191,08, ou, R\$ 38,22 por unidade fabricada da placa, sem considerar ainda os valores gastos com impostos de importação. Como discutido anteriormente, a alta do dólar afeta significativamente na disponibilidade de insumos para a confecção de produtos eletrônicos.

Figura 55 – Carregador balanceador para baterias do tipo *Li-Po*



Fonte: <www.filipeflop.com>.

Outro elemento que pode ser adquirido é um carregador adequado para as baterias do tipo *Li-Po*, sendo recomendada a compra de modelos do tipo balanceador, que realiza um carregamento proporcional apropriado para células que serão utilizadas em conjunto. Durante a pesquisa de preços, os produtos deste tipo apresentaram valores dentro da margem de preço vista na Tabela 10, todavia, apenas um carregador

pode ser suficiente para carregar a bateria de vários dispositivos, diluindo também o custo final por cada conjunto construído. O modelo mais comum encontrado para carregadores deste tipo é o B6 Mini, que possui aspecto físico igual ao visto na Figura 55.

Um cenário de custo diluído também se aplica ao caso do gravador *ST-LINK V2*, utilizado para escrever os programas necessários ao funcionamento do sistema embarcado, podendo ser compartilhado entre vários *kits*. Embora seja possível adquirir estes elementos por valores mais baixos em lojas internacionais, os valores apresentados na Tabela 10 para lojas brasileiras ainda são viáveis por conta da acessibilidade de entrega. Na Figura 56 é possível visualizar o gravador comumente empregado na escrita de microcontroladores da família STM.

Figura 56 – Gravador para plataformas STM



Fonte: <www.eletrogate.com>.

5.3 Comparação com sistemas de referência

Embora a análise individual dos custos do projeto já seja capaz de trazer uma avaliação interessante do desenvolvimento da plataforma, é importante comparar esse produto final desenvolvido com as soluções comerciais que foram apresentadas anteriormente e serviram como inspiração direta para a idealização das funcionalidade do sistema confeccionado. Por meio da Tabela 11, é possível visualizar de maneira breve e genérica a comparação dos preços verificados para as plataformas comerciais avaliadas em relação ao valor final de custo obtido para este projeto.

Com foco no critério comparativo, foram tomados os modelos de *hardware* que apresentam funções e estruturas similares ao protótipo idealizado por este projeto, focando nas plataformas que pudessem competir e preencher funcionalidades muito parecidas. Como previsto anteriormente, a comparação direta entre o preço das solu-

ções de cunho comercial e o custo final do robô, apresentado neste estudo, trouxe um cenário muito favorável à implementação proposta, notando que foi possível atingir um custo de confecção abaixo dos valores encontrados para a aquisição destes produtos em mercado nacional. Voltando à discussão feita nos primeiros capítulos, grande parte desses custos são reflexo direto de políticas de importação que imputam uma série de taxações sobre os produtos eletrônicos e inviabilizam grande parte das transações internacionais nesse nicho de mercado, principalmente em escala reduzida de compra, direcionadas ao usuário final.

Tabela 11 – Comparativo de custos do projeto com os kits de referência

<i>Kit</i>	Preço (US\$)	Preço (R\$)
<i>LEGO Mindstorms EV3®</i>	349,99	2.999,99*
<i>TI-RSLK®</i>	199,00	-
<i>mBot Ranger®</i>	127,49	1.497,90*
<i>Kit Robótica Arduino Genérico**</i>	-	299,90
Este projeto	-	362,57***

Fonte: Autoria Própria.

(*) Valores de compra encontrados no mercado brasileiro

(**) Kit de robótica voltado à utilização da plataforma *Arduino Uno*, acessível em:
<www.eletragate.com/kit-arduino-robotica>

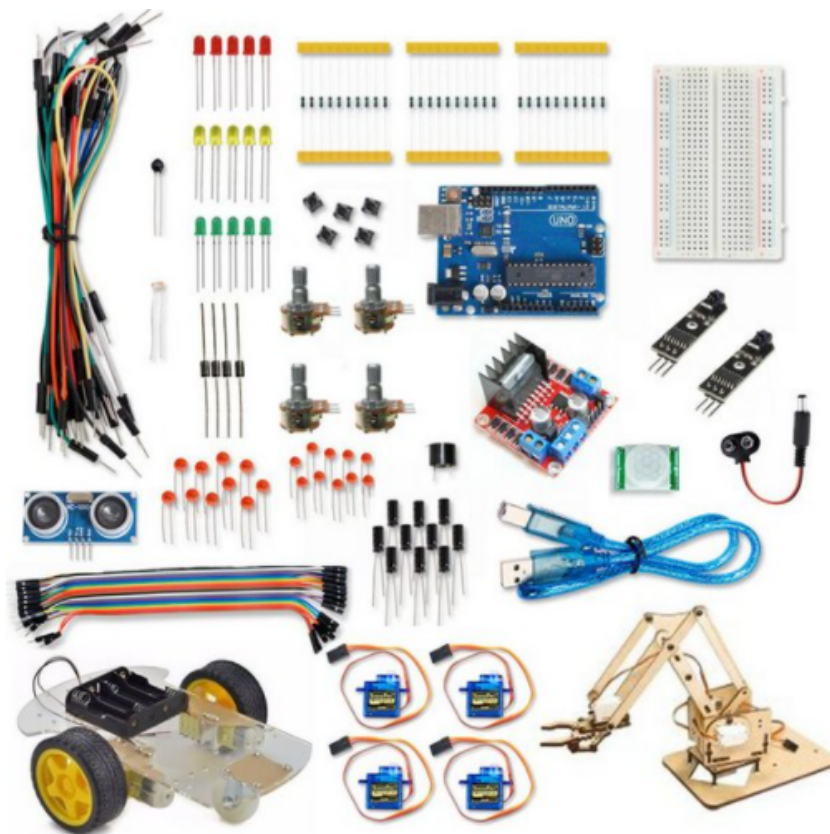
(***) Custo dos componentes somado à estimativa de custo individual da placa confeccionada industrialmente

Em termos de funcionalidade, embora apresentando um comportamento ainda restrito, o conjunto de soluções proposto permite a implementação de funções muito próximas às vistas nos kits comerciais de introdução à robótica, com o uso de sensores para a aquisição de dados e acionamento completo dos motores, abrindo ao usuário a possibilidade de desenvolver seus próprios estudos sobre os princípios da robótica. Além destes critérios, desde sua concepção, o estudo redigido aqui procura evidenciar a possibilidade de personalização de cada uma das partes adicionadas e, por meio de ajustes na placa de circuito impresso e do programa de automação, garantir ao usuário um entendimento básico do processo de projeto para que se sinta encorajado a realizar suas próprias alterações e criar, de maneira didática, o seu próprio robô.

Ainda, como critério de comparação para um cenário mais acessível no mercado interno, ao fazer a pesquisa de custos para os componentes discretos necessários ao projeto, alguns conjuntos prontos para desenvolvimento de aplicações voltadas ao estudo de princípios da robótica foram verificadas, normalmente direcionadas ao uso da plataforma *Arduino* e seus diferentes modelos de microcontroladores. Nesses kits, alguns componentes básicos são adicionados e permitem a implementação de soluções simplificadas capazes de despertar o interesse por essa área do conhecimento.

O quarto item da Tabela 11 representa um destes conjuntos, selecionado para esta avaliação por apresentar características técnicas muito próximas ao resultado almejado para este estudo e, principalmente, por emular o formato de um carrinho, conforme visto na Figura 57.

Figura 57 – Kit básico de robótica para plataformas *Arduino*



Fonte: <www.eletrogate.com>.

Avaliando a opção dos *kits* genéricos, estes podem ser adquirido para a construção de um robô simples, com características próximas às definidas para este estudo, como sensoramento de distância, modo seguidor de linha, sensor de presença e até um acionamento adequado aos motores do robô. Uma vantagem, específica a este *kit* genérico apresentado, é a presença do braço mecânico controlável, uma solução interessante que também pode ser adaptada e personalizada ao conjunto apresentado neste projeto.

Entretanto, mesmo com um valor um pouco mais elevado no comparativo bruto, a solução aqui proposta apresenta um maior número de sensores, a possibilidade de comunicação via *bluetooth* e, ainda, a presença de quatro motores para a mobilidade do sistema, em contraponto a apenas dois motores do *kit* genérico, todos estes, componentes que acarretam um aumento significativo de custo ao sistema. Desse modo, o *kit* didático proposto ainda se mostra vantajoso financeiramente, trazendo

consigo uma quantidade maior de funcionalidades ao custo de um acréscimo de pouco impacto no montante final. Ainda, é possível ressaltar que a personalização destes elementos pode trazer uma redução dos custos envolvidos, dependendo diretamente da intenção do usuário em relação ao protótipo final.

Outro ponto de vantagem do *kit* proposto é a existência de uma placa de circuito impresso dedicada, construída especificamente para o propósito idealizado, situação não compreendida pelo *kit* genérico comercial, que conta apenas com uma matriz de contatos para a montagem completa do sistema. Embora uma matriz de contatos seja acessível e didática, circuitos montados por meio dessas conexões apresentam comportamento mais instável e organização dificultada, com excesso de fios e conexões improvisadas, que podem se soltar com o movimento do robô. Para um melhor desempenho, o uso de placas de circuito impresso é recomendada para qualquer circuito que se deseje implementar.

6 CONSIDERAÇÕES FINAIS

6.1 Análise dos resultados

O desenvolvimento deste trabalho baseou-se na tentativa de apresentar um documento de referência para a montagem de um *kit* didático, voltado ao ensino de robótica e preocupado com o custo e a acessibilidade das ferramentas e insumos empregados em sua construção.

Para entender a realidade do ensino no Brasil, principalmente das instituições de ensino superior, e sua relação cotidiana com a tecnologia, o estudo iniciou com uma pesquisa de exploração, tentando elencar as dificuldades encontradas por alunos e professores na busca pelo conhecimento e no acesso à informação. Por meio dessas discussões, foi possível identificar que os fatores econômicos causam impacto significativo nas decisões de ingresso e permanência de estudantes em seus cursos de interesse e, aliados à pouca atratividade presenciada em alguns espaços do ambiente escolar, são responsáveis por alargar as barreiras socioeconômicas vivenciadas.

Em contraponto, o estudo permitiu identificar ações em prol de melhores condições de ensino que já são executadas em âmbito nacional, descrevendo abordagens utilizadas para promover o ensino multidisciplinar e uso lúdico da tecnologia. Dentre estes movimentos encontra-se a robótica educacional, método de ensino que utiliza conceitos pedagógicos com histórico de sucesso, aliando o aprendizado como experimentação à revolução tecnológica que ocorre em escala mundial.

Por meio destes dados encontrados, foi identificado como relevante o projeto de um *kit* didático, alinhado ao conceito de robótica livre cunhado por César (2013), que se preocupa com o custo de implementação e o uso de ferramentas de livre acesso, pensando em meios de incentivar a democratização do ensino. O primeiro passo na busca por uma implementação viável deu-se pela identificação das tecnologias atualmente disponíveis em âmbito comercial, caracterizando suas funcionalidades e tomando-as como inspiração para o desenvolvimento de uma solução que pudesse competir em potencial educativo e custo de produção.

A análise de mercado permitiu perceber que a parte mais relevante de um circuito voltado ao estudo de robótica é seu núcleo de controle, comumente implementado por plataformas microcontroladas. Tomando as especificações pretendidas para o protótipo em construção, iniciou-se a procura por um microcontrolador que fosse compatível com as necessidades estabelecidas e, ainda, apresentasse condições de aquisição vantajosas, respeitando o caráter de baixo custo, definido como essencial. Algumas plataformas diferentes foram avaliadas, resultando na escolha de uma delas, a *STM32F103C8TX*, de custo modesto, facilmente encontrada no mercado nacional e compatível com linguagem de programação amparada por vasta documentação de

implementações.

Como resultado, a definição dos critérios mínimos para o desenvolvimento de uma plataforma didática pôde ser estabelecida, com a seleção dos sensores, elementos mecânicos e circuitos auxiliares que fariam parte do sistema, tentando imitar o máximo de funcionalidades desempenhadas pelos *kits* comerciais estudados, na tentativa de substituí-los. As especificações do projeto serviram como base para a implementação do circuito desejado, pensada para ser executada em três diferentes etapas.

A primeira parte do conjunto a ser desenvolvida, foi o projeto de uma placa de circuito impresso, responsável por realizar as conexões de todos os elementos do circuito e servir como base mecânica para a estrutura de componentes do sistema embarcado. Este projeto contou com um diagrama de esquemático próprio, que levou em conta o condicionamento de todas as topologias de circuito e dos módulos associados ao *kit*, garantindo que as peculiaridades de cada um fosse atendida. Depois dessas distribuições, foi possível estabelecer um *layout* adequado para a construção física da placa, desenvolvido com cuidado em relação aos comportamentos elétricos do sistema e levando em conta uma série de boas práticas de desenho de PCI.

Com o *hardware* finalizado, foi possível iniciar o desenvolvimento do programa que será gravado no microcontrolador e fará o gerenciamento de todas as funcionalidades previstas para o *kit* de robótica. Nesta etapa, foram configurados os parâmetros de todas as interfaces de comunicação, necessárias para a integração dos módulos de sensoramento, desenvolvida a rotina de controle para o acionamento dos motores do carrinho e também implementadas as condições de temporização do sistema. O resultado dessas interações foi um programa funcional que cumpriu as expectativas propostas para a automação do conjunto.

Por conta da capacidade do robô em comunicar-se com dispositivos externos por meio de interface sem fio, foi realizada uma busca por ferramentas acessíveis para a construção de aplicativos móveis capazes de conectar-se e trocar informações com o sistema embarcado. Uma das plataformas de desenvolvimento encontradas, a *App Inventor*, se mostrou bastante interessante e viável, permitindo a construção de uma aplicação simples, que serviu como prova de conceito e um controle externo para a aplicação do robô. O resultado foi um aplicativo de demonstração, com interface visual personalizada, capaz de enviar comandos básicos de controle ao protótipo e receber eventos com a leitura dos sensores conectados ao carrinho.

Com a conclusão das soluções planejadas, uma análise de custos permitiu avaliar o impacto causado pelas decisões tomadas ao longo do desenvolvimento, principalmente no aspecto financeiro, garantindo que a acessibilidade prevista na pesquisa inicial fosse cumprida. Pensando nas expectativas previstas no início do desenvolvimento e comparando com o cenário verificado ao final de toda a execução,

é inevitável perceber que parte do processo se perdeu por conta da grande limitação de tempo e recursos que foi instaurada com o fechamento do ambiente acadêmico de produção em laboratório do IFSC tendo em vista o cumprimento regras de quarentena contra a pandemia de COVID-19, notando que estas afetaram significativamente o cronograma das atividades planejadas.

Entretanto, avaliando por outro ponto de vista e estabelecendo critérios diferenciados de análise, o projeto desenvolvido conseguiu cumprir em seu estudo teórico o desenvolvimento da maior parte das funcionalidades esperadas, principalmente levando em conta o tempo disponível para a execução destas tarefas, além de cumprir a maioria dos requisitos estabelecidos para a aplicação dos conceitos inspirados nos conjuntos de robótica encontrados em comercialização no mercado de produtos eletrônicos.

Desse modo, é possível dizer que o projeto cumpriu os requisitos estabelecidos e resultou em um documento interessante sobre o processo de confecção, deixando ainda algumas possibilidades de continuação do trabalho, entretanto, apresetando um cenário promissor para a implementação previamente imaginada.

Para a persistência das informações e garantia do acesso facilitado dos estudos e das implementações propostas, foi criado um repositório na plataforma *GitHub*, visando a disponibilização de todos os documentos e informações utilizadas na implementação do conjunto, contando inclusive com os projetos vinculados ao *hardware*, *software* e *firmware* do sistema e também ao desenvolvimento do aplicativo móvel para conexão com o robô. Este repositório pode ser encontrado através do *link* <www.github.com/joaov28/kit_didatico_robotica>. Com os resultados obtidos, a expectativa para os usos futuros das informações acumuladas é de que estas possam ser aplicadas em diversas abordagens de ensino prático e que inspirem a construção de outras soluções personalizadas, incentivando a disseminação de ideias que tentam minimizar as dificuldades de ensino enfrentadas na realidade nacional.

6.2 Propostas de projetos futuros

Como mencionado anteriormente, a limitação de acesso a ferramentas apropriadas e a redução do tempo disponível para execução do estudo criaram barreiras para a implementação de algumas das funcionalidades previstas nas fases iniciais de desenvolvimento criativo da solução, obrigando um corte e remoção destas funções adicionais, que refinariam o projeto como um todo. Desse modo, é importante apresentar as adições previstas anteriormente para que estas possam ser executadas em interações futuras, melhorando o funcionamento e a usabilidade do protótipo.

Uma das primeiras tarefas a ser obrigatoriamente realizada é a efetiva implementação e montagem das partes do sistema, garantindo que todo o conjunto funciona corretamente, verificando a plena interação entre os componentes de *hardware* e o

programa em *software*. Nesse tópico, o foco principal deve ser a confecção da placa de circuito impresso e o teste dos circuitos que dependem de maior potência de alimentação, partes que não puderam ser testadas em nenhum trecho do desenvolvimento.

Algumas outras melhorias podem ser executadas para encontrar uma condição de funcionamento mais rebuscada, principalmente no *software* e no aplicativo móvel criado, que foram desenvolvidos de modo a atingir o mínimo estado funcional, deixando passar algumas otimizações e métodos de implementação mais robustos. Um exemplo de melhoria é a troca da execução de determinadas interações em *software* para que passem a utilizar interrupções de *hardware*, que apresentam vantagem ao não atrapalhar o funcionamento geral da aplicação, ajudando assim a diminuir o desperdício de processamento verificado em algumas tarefas.

Pensando no critério de redução do custo associado na reprodução do sistema descrito, como discutido nos resultados do projeto, um dos elementos que impacta nesse montante é a parte mecânica, que serve como base na construção do *kit*. Uma opção viável é o uso de impressões 3D para a montagem de protótipos que, com a disponibilidade adequada de insumos, permite uma redução de custos e uma possibilidade de personalização interessante, garantindo ao usuário a plena decisão sobre o aspecto final do sistema montado. Para isso, é indicada a criação de um modelo 3D padrão que possa ser disponibilizado juntamente com o *layout* da placa de circuito impresso desenvolvida aqui, deixando o processo ainda mais artesanal.

Ainda, inspirado por esse caráter de construção gradativa do conjunto e pensando na personalização do *kit*, após garantir que todos os subsistemas funcionam e interagem corretamente entre si, é interessante a criação e disponibilização de um manual prático, visual e intuitivo, que apresente um tutorial simplificado de montagem do circuito e identifique o posicionamento adequado dos sensores, as conexões mecânicas e os detalhes de execução do projeto.

No aspecto didático e com foco na iniciação ao mundo da programação e da robótica, uma preocupação futura pode levar em conta a inserção de usuários de diferentes níveis de experiência, não necessariamente montando o robô, mas fazendo o uso da plataforma já montada para a execução de tarefas simples. Para cumprir esse requisito, é interessante o uso de plataformas de tradução de linguagens de programação voltadas a abstrações em sistemas de blocos, como a chamada *Blockly*, que adiciona a possibilidade de criar rotinas de execução por meio de programação visual, traduzindo-as posteriormente em diferentes linguagens de programação, tornando o processo mais intuitivo e simplificado àqueles que não possuem contato extenso com esse universo.

REFERÊNCIAS

- ANJITHA, G. *Obstacle Avoidance Robot Using FPGA, Mini Project Report*. 2016. Disponível em: <https://www.researchgate.net/publication/341480237_OBSTACLE_AVOIDANCE_ROBOT_USING_FPGA>. Acesso em: 29 de setembro de 2020. Citado na página 100.
- AROCA, R. V. *Plataforma robótica de baixíssimo custo para robótica educacional*. Tese (Doutorado) — Programa de Pós-graduação em Engenharia Elétrica e Computação, Universidade Federal do Rio Grande do Norte, Natal, 2012. Disponível em: <https://repositorio.ufrn.br/jspui/bitstream/123456789/15206/1/RafaelVA_DISSERT.pdf>. Acesso em: 29 de setembro de 2020. Citado na página 39.
- BASTOS, M. O. *A informática a serviço da construção do conhecimento na tarefa docente*. Dissertação (Mestrado) — Programa de Pós-graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, 2002. Citado na página 23.
- BOSCH SENSORTEC. *BMP180 - Digital Pressure Sensor*. 2013. Disponível em: <<https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf>>. Acesso em: 30 de setembro de 2020. Citado na página 65.
- CAMPO, A. B. Robótica mole: o estado da arte e um roteiro para *makers*. In: _____. *Robótica Educacional - Experiências Inovadoras na Educação Brasileira*. Porto Alegre: Penso Editora Ltda., 2020. ISBN 9788584291892. Citado na página 49.
- CONTROLLERS TECH. *HC-SR04 Ultrasonic sensor and STM32*. 2019. Disponível em: <<https://controllerstech.com/hc-sr04-ultrasonic-sensor-and-stm32>>. Acesso em: 30 de setembro de 2020. Citado na página 102.
- CÉSAR, D. R. *Robótica pedagógica livre: uma alternativa metodológica para a emancipação sociodigital e a democratização do conhecimento*. Tese (Doutorado) — Faculdade de Educação, Universidade Federal da Bahia, Salvador, 2013. Disponível em: <https://repositorio.ufba.br/ri/bitstream/ri/16087/1/Tese_revisada_final.pdf>. Acesso em: 29 de setembro de 2020. Citado 2 vezes nas páginas 24 e 131.
- CÉSAR, D. R. Robótica pedagógica livre e artefatos cognitivos na/para a construção do conhecimento. In: _____. *Robótica Educacional - Experiências Inovadoras na Educação Brasileira*. Porto Alegre: Penso Editora Ltda., 2020. ISBN 9788584291892. Citado 3 vezes nas páginas 24, 49 e 51.
- FREGONESIS, J. G. P. *Estudo do desempenho acadêmico nos cursos de graduação dos centros de ciências exatas e de tecnologia da Universidade Estadual de Maringá: período 1995-2000*. Dissertação (Mestrado) — Programa de Pós-graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, 2002. Citado na página 19.
- GEMAA. *O que são ações afirmativas?* s.d. Página institucional do Grupo de Estudos Multidisciplinares de Ações Afirmativas. Disponível em: <<http://gemaa.iesp.uerj.br/o-que-sao-acoes-afirmativas>>. Acesso em: 29 de setembro de 2020. Citado na página 18.

- GIRJAU, G. *BMP180 C library using STM32 I2C HAL driver*. 2019. Disponível em: <<https://gist.github.com/ggirjau/369d1520a21c7cf41413b00352446d61>>. Acesso em: 30 de setembro de 2020. Citado na página 95.
- INVENSENSE. *MPU-6000 and MPU-6050 Product Specification*. 2012. Disponível em: <<http://www.haoyuelectronics.com/Attachment/GY-521/mpu6050.pdf>>. Acesso em: 30 de setembro de 2020. Citado na página 65.
- ITEAD STUDIO. *HC-05- Bluetooth to Serial Port Module*. 2010. Disponível em: <<https://datasheetspdf.com/pdf-file/1418730/ITead/HC-05/1>>. Acesso em: 30 de setembro de 2020. Citado na página 66.
- KONSTANTIN, B. *STM32 HAL library for GY-521 (MPU6050) with Kalman filter*. 2020. Disponível em: <<https://github.com/leech001/MPU6050>>. Acesso em: 30 de setembro de 2020. Citado na página 95.
- LEGO, G. *LEGO Mindstorms EV3 - User Guide*. [S.l.], 2013. Disponível em: <<https://fccid.io/NPI95646/User-Manual/User-manual-part-1-2083077.pdf>>. Acesso em: 29 de setembro de 2020. Citado na página 30.
- MENDES JUNIOR, A. F. Uma análise da progressão dos alunos cotistas sob a primeira ação afirmativa brasileira no ensino superior: o caso da universidade do estado do rio de janeiro. *Ensaio: Avaliação e Políticas Públicas em Educação*, v. 22, n. 82, p. 31–56, 2014. Citado na página 18.
- MINISTÉRIO DA ECONOMIA Receita Federal. *Tabela de Incidência do Imposto sobre Produtos Industrializados (TIPI)*. 2017. Disponível em: <<http://receita.economia.gov.br/acesso-rapido/legislacao/documentos-e-arquivos/tipi-1.pdf/view>>. Acesso em: 30 de setembro de 2020. Citado na página 21.
- MINISTÉRIO DA ECONOMIA Receita Federal. *Regime de Tributação Simplificada - RTS*. s.d. Disponível em: <<https://receita.economia.gov.br/orientacao/aduaneira/manuais/remessas-postal-e-expressa/topicos/tributacao>>. Acesso em: 30 de setembro de 2020. Citado na página 21.
- MINISTÉRIO DA ECONOMIA Receita Federal. *Simulador do Tratamento Tributário e Administrativo das Importações*. s.d. Disponível em: <<http://www4.receita.fazenda.gov.br/simulador/>>. Acesso em: 30 de setembro de 2020. Citado na página 21.
- MIRANDA, L. C. d.; SAMPAIO, F. F.; BORGES, J. A. d. S. Robofácil: Especificação e implementação de um kit de robótica para a realidade educacional brasileira. *Revista Brasileira de Informática na Educação*, v. 18, n. 3, p. 46–58, 2010. Citado na página 40.
- MIT. *Building Apps with many Screens*. c. 2012–2015. Disponível em: <<http://ai2.appinventor.mit.edu/reference/other/manyscreens.html>>. Acesso em: 30 de setembro de 2020. Citado na página 108.
- MORGAN, E. *HC-SR04 Ultrasonic Sensor*. 2014. Disponível em: <<https://datasheetspdf.com/pdf-file/1380136/ETC/HC-SR04/1>>. Acesso em: 30 de setembro de 2020. Citado 2 vezes nas páginas 67 e 100.

MPS. *Datasheet: MP1584 Step-Down Converter*. 2011. Disponível em: <<https://storage.googleapis.com/baudaeletronicadatasheet/MP1584EN.pdf>>. Acesso em: 29 de setembro de 2020. Citado na página 62.

OBR. *O que é a OBR (Olimpíada Brasileira de Robótica)?* c. 2020. Disponível em: <<http://www.obr.org.br/o-que-e-a-obr/>>. Acesso em: 30 de setembro de 2020. Citado na página 26.

ON SEMICONDUCTOR. *Datasheet: BD136/138/140 PNP Epitaxial Silicon Transistor*. 2000. Disponível em: <<https://www.mouser.com/datasheet/2/149/BD140-888626.pdf>>. Acesso em: 29 de setembro de 2020. Citado na página 63.

ON SEMICONDUCTOR. *Datasheet: BD135/137/139 NPN Epitaxial Silicon Transistor*. 2017. Disponível em: <<https://www.onsemi.com/pub/Collateral/BD139-D.PDF>>. Acesso em: 29 de setembro de 2020. Citado na página 63.

PUGLIESE, G. O. *Os modelos pedagógicos de ensino de ciências em dois programas educacionais baseados em STEM (Science, Technology, Engineering and Mathematics)*. Dissertação (Mestrado) — Universidade Estadual de Campinas, Campinas, 2017. Citado na página 25.

ROBOCUP BRAZIL. *Competição Brasileira de Robótica*. c. 2019. Disponível em: <<http://www.cbrobotica.org/index.php/o-evento/>>. Acesso em: 30 de setembro de 2020. Citado na página 26.

SACCARO, A.; FRANÇA, M. T. A.; JACINTO, P. d. A. Fatores associados à evasão no ensino superior brasileiro: um estudo de análise de sobrevivência para os cursos das áreas de ciência, matemática e computação e de engenharia, produção e construção em instituições públicas e privadas. *Estudos Econômicos*, v. 49, n. 2, p. 337–373, 2019. Citado 2 vezes nas páginas 17 e 18.

SHANGAI TX ELECTRONICS. *Datasheet: TX6211B Series - High Speed Low Noise LDO*. 2015. Disponível em: <http://www.txsemi.com/Upload/TX6211B_V12-11264794020.pdf>. Acesso em: 29 de setembro de 2020. Citado na página 57.

SILVA, F. I. da; SCHERER, D. Praxedes: protótipo de um kit educacional de robótica baseado na plataforma arduino. *EaD & Tecnologias Digitais na Educação*, v. 1, n. 1, p. 44–56, 2013. Citado na página 40.

SILVA, H. F. d. *Robótica educacional como recurso pedagógico fomentador do Itramento científico de alunos da rede pública de ensino na cidade do Recife*. Dissertação (Mestrado) — Programa de Pós-graduação em Educação Matemática e Tecnológica, Universidade Federal de Pernambuco, Recife, 2018. Citado na página 27.

ST MICROELECTRONICS. *Datasheet: STM32F103x8 and STM32F103xB*. 2015. Disponível em: <<https://www.st.com/resource/en/datasheet/stm32f103c8.pdf>>. Acesso em: 30 de setembro de 2020. Citado na página 57.

STARKE, R. A. *Github: Documentação para construção de hardware para um carrinho microcontrolado*. c. 2020. Disponível em: <<https://github.com/xtarke/avrlibs/tree/master/shields>>. Acesso em: 30 de setembro de 2020. Citado na página 57.

TDK CORPORATION. *Piezoelectric Buzzers - Pin Terminal/Lead Without Oscillator Circuit*. 2011. Disponível em: <https://www.mouser.com/datasheet/2/400/ef532_ps-13444.pdf>. Acesso em: 30 de setembro de 2020. Citado na página 73.

TEXAS INSTRUMENTS. *Datasheet: LMx93-N, LM2903-N Low-Power, Low-Offset Voltage, Dual Comparators*. 2018. Disponível em: <<https://www.ti.com/lit/ds/symlink/lm393-n.pdf>>. Acesso em: 30 de setembro de 2020. Citado na página 69.

TEXAS INSTRUMENTS. *Texas Instruments Robotics System Learning Kit - The Maze Edition*. [S.l.], 2018. Disponível em: <<https://university.ti.com/en/faculty/ti-robotics-system-learning-kit/ti-robotics-system-learning-kit/curriculum-design-launch>>. Acesso em: 29 de setembro de 2020. Citado na página 33.

VALLANCE, M. et al. *LEGO Mindstorms™ for informed metrics in virtual worlds. People and Computers XXIII Celebrating People and Technology - Proceedings of HCI*, p. 31–56, 2009. Citado na página 29.

VISHAY SEMICONDUCTORS. *Datasheet: TCRT5000, TCRT5000L*. 2019. Disponível em: <<https://www.vishay.com/docs/83760/tcrt5000.pdf>>. Acesso em: 29 de setembro de 2020. Citado 2 vezes nas páginas 68 e 69.

WATTERS, A. *Lego Mindstorms: A History of Educational Robots*. 2015. Medium. Disponível em: <<https://medium.com/the-history-of-the-future-of-education/lego-mindstorms-a-history-of-educational-robots-cc323766e8c1>>. Acesso em: 29 de setembro de 2020. Citado 2 vezes nas páginas 28 e 30.

ZILLI, S. d. R. *A Robótica Educacional no Ensino Fundamental: Perspectivas e Prática*. Dissertação (Mestrado) — Programa de Pós-graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, 2004. Citado 2 vezes nas páginas 22 e 23.

APÊNDICES

APÊNDICE A – IMPLEMENTAÇÕES DE CÓDIGO PARA O SISTEMA EMBARCADO

Código A.1 – Estrutura básica de um *timer*

```

1 typedef uint32_t time;
2
3 typedef enum {ONE_TIME_MODE, CONTINUOUS_MODE} modes_t;
4
5 typedef struct
6 {
7     time tick_count;
8     time timeout;
9     modes_t timer_mode;
10 } timers_t;

```

Código A.2 – Implementação de temporizações não-bloqueantes por software

```

11 void Timer_Init(timers_t *timer, modes_t mode)
12 {
13     timer->start_tick = 0;
14     timer->timeout = 0;
15     timer->timer_mode = mode;
16 }
17
18 void Timer_Start(timers_t *timer, time curr_tick, time interval)
19 {
20     timer->start_tick = (curr_tick? curr_tick : 1);
21     timer->timeout = interval;
22 }
23
24 void Timer_Stop(timers_t *timer)
25 {
26     timer->start_tick = 0;
27 }
28
29 time timeDiff(time curr_tick, time prev_tick)
30 {
31     time diff;
32
33     if(curr_tick >= prev_tick) { diff = curr_tick - prev_tick;}
34     else { diff = (0xFFFFFFFF - prev_tick) + curr_tick;}
35
36     return diff;
37 }
38
39 bool Timer_Timeout(timers_t *timer, time curr_tick)

```

```
40 {
41     const bool t_out = (timer->start_tick && timeDiff(curr_tick, timer->start_tick) >
42         timer->timeout);
43     if(t_out)
44     {
45         Timer_Stop(timer);
46         if(timer->timer_mode == CONTINUOUS_MODE)
47             Timer_Start(timer, curr_tick, timer->timeout);
48     }
49     return t_out;
50 }
```

Código A.3 – Exemplo de uso de um timer

```
51 timers_t debug_timer;
52
53 Timer_Init(&debug_timer, CONTINUOUS_MODE);
54 Timer_Start(&debug_timer, HAL_GetTick(), 5000);
55
56 while(1)
57 {
58     if(Timer_Timeout(&debug_timer, HAL_GetTick()))
59         print_debug();
60 }
```

Código A.4 – Função de controle da direção de deslocamento do robô

```
61 extern TIM_HandleTypeDef htim1;
62
63 direction_t direction = STOP;
64 uint8_t speed = 0;
65
66 void Motor_ChangeDirection( void )
67 {
68     int pwm1 = 0;
69     int pwm2 = 0;
70
71     switch( direction )
72     {
73         case FORWARD:
74         {
75             pwm1 = 5000 + (speed * 1000);
76             pwm2 = 5000 + (speed * 1000);
77
78             __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, pwm1);
79             __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_2, pwm2);
```

```
80     HAL_TIMEx_PWMN_Stop(&htim1,TIM_CHANNEL_1);
81     HAL_TIMEx_PWMN_Stop(&htim1,TIM_CHANNEL_2);
82     HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_1);
83     HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_2);
84     break;
85 }
86
87 case BACKWARD:
88 {
89     pwm1 = 5000 + (speed * 1000);
90     pwm2 = 5000 + (speed * 1000);
91
92     __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1,pwm1);
93     __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_2,pwm2);
94     HAL_TIM_PWM_Stop(&htim1,TIM_CHANNEL_1);
95     HAL_TIM_PWM_Stop(&htim1,TIM_CHANNEL_2);
96     HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_1);
97     HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_2);
98     break;
99 }
100
101 case LEFT:
102 {
103     pwm1 = 5000 + (speed * 1000);
104     pwm2 = 1000 + (speed * 500);
105
106     __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1,pwm1);
107     __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_2,pwm2);
108     HAL_TIMEx_PWMN_Stop(&htim1,TIM_CHANNEL_1);
109     HAL_TIMEx_PWMN_Stop(&htim1,TIM_CHANNEL_2);
110     HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_1);
111     HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_2);
112     break;
113 }
114
115 case RIGHT:
116 {
117     pwm1 = 1000 + (speed * 500);
118     pwm2 = 5000 + (speed * 1000);
119
120     __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1,pwm1);
121     __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_2,pwm2);
122     HAL_TIMEx_PWMN_Stop(&htim1,TIM_CHANNEL_1);
123     HAL_TIMEx_PWMN_Stop(&htim1,TIM_CHANNEL_2);
124     HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_1);
125     HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_2);
126     break;
```

```

127     }
128
129     case STOP:
130     {
131         HAL_TIM_PWM_Stop(&htim1,TIM_CHANNEL_1);
132         HAL_TIM_PWM_Stop(&htim1,TIM_CHANNEL_2);
133         HAL_TIMEx_PWMN_Stop(&htim1,TIM_CHANNEL_1);
134         HAL_TIMEx_PWMN_Stop(&htim1,TIM_CHANNEL_2);
135         break;
136     }
137 }
138 }

```

Código A.5 – Manipulação da aquisição de dados pelo ADC

```

139 uint32_t batt_1 = 0;
140 uint32_t batt_2 = 0;
141
142 void ADC_ReadData( void )
143 {
144     uint32_t adc1_value = 0;
145     uint32_t adc2_value = 0;
146
147     HAL_ADC_Start(&hadc1);
148     HAL_ADC_PollForConversion(&hadc1, 50);
149     adc1_value = HAL_ADC_GetValue(&hadc1);
150     HAL_ADC_PollForConversion(&hadc1, 50);
151     adc2_value = HAL_ADC_GetValue(&hadc1);
152     HAL_ADC_Stop(&hadc1);
153
154     adc1_value = (adc1_value*3300)/4095;
155     adc2_value = (adc2_value*3300)/4095;
156
157     batt_2 = (adc2_value*15)/10;
158     batt_1 = (adc1_value*3) - batt_2;
159 }

```

Código A.6 – Recepção de pacotes do protocolo UART

```

160 char rxBuffer[32] = {0};
161 char received_byte = 0;
162
163 static uint8_t get_number( uint8_t ascii )
164 {
165     switch(ascii)
166     {
167     case '0' ... '9':

```

```
168     return ascii - '0';
169     break;
170     case 'A' ... 'F':
171         return ascii - 'A' + 0x0A;
172         break;
173     }
174     return 0;
175 }
176
177 void UART_ReadData( void )
178 {
179     if(HAL_UART_Receive(&huart2, (uint8_t *)&received_byte, 1, 1) == HAL_OK)
180     {
181         UART_RXHandle( received_byte );
182     }
183 }
184
185 void UART_RXHandle( uint8_t received_byte )
186 {
187     static uint8_t i = 0, j = 0, start_rec = 0x00;
188
189     if(received_byte == '@')
190     {
191         start_rec = 0x01;
192         i = 0; j = 0;
193         memset(rxBuffer, 0x00, sizeof(rxBuffer));
194         return;
195     }
196
197     if(received_byte == '!')
198     {
199         start_rec = 0x00;
200         UART_CommandsHandle();
201     }
202
203     if(start_rec)
204     {
205         uint8_t is_even = (j % 2)? 0 : 1;
206         if(is_even)
207             rxBuffer[i] |= ((get_number(received_byte) & 0x0F) << 4);
208         else
209         {
210             rxBuffer[i] |= (get_number(received_byte) & 0x0F);
211             i++;
212         }
213         j++;
214     }
```

```
215 }
```

Código A.7 – Processamento dos comandos recebidos

```
216 void UART_CommandsHandle( void )
217 {
218     switch(rxBuffer[0])
219     {
220     case DIRECTION:
221         direction = rxBuffer[1];
222         Motor_ChangeDirection();
223         break;
224     case LEDS:
225         LED_ChangeRoutine(rxBuffer[1]);
226         break;
227     case SENSORS_UPDATE:
228         Sensors_Update();
229         break;
230     }
231 }
```

Código A.8 – Exemplo de utilização da transmissão de um evento

```
232 void Battery_UpdateStatus( void )
233 {
234     char message[16] = {0};
235     uint8_t battery_status = 0;
236
237     if(batt_1 > 3600 && batt_2 > 3600)
238     {
239         battery_status = 0x03;
240         HAL_GPIO_WritePin(D6_GPIO_Port, D6_Pin, GPIO_PIN_RESET);
241     }
242     else if(batt_1 > 3400 && batt_2 > 3400)
243     {
244         battery_status = 0x02;
245         HAL_GPIO_WritePin(D6_GPIO_Port, D6_Pin, GPIO_PIN_RESET);
246     }
247     else if(batt_1 > 3200 && batt_2 > 3200)
248     {
249         battery_status = 0x01;
250         HAL_GPIO_WritePin(D6_GPIO_Port, D6_Pin, GPIO_PIN_SET);
251     }
252     else
253     {
254         battery_status = 0x00;
255         HAL_GPIO_WritePin(D6_GPIO_Port, D6_Pin, GPIO_PIN_SET);

```

```

256     }
257
258     snprintf(message, 16, "%.2x%d!", BATTERY_STATUS, battery_status);
259     HAL_UART_Transmit(&huart2, (uint8_t*)message, sizeof(message), 50);
260 }

```

Código A.9 – Rotina de contagem de tempo em microssegundos

```

261 extern TIM_HandleTypeDef htim3;
262
263 void delay_us( uint16_t interval )
264 {
265     __HAL_TIM_SET_COUNTER(&htim3, 0);
266     while((uint16_t)__HAL_TIM_GET_COUNTER(&htim3) < interval);
267 }

```

Código A.10 – Rotinas de interação com o módulo HC-SR04

```

268 extern TIM_HandleTypeDef htim3;
269
270 uint32_t IC_Val1 = 0;
271 uint32_t IC_Val2 = 0;
272 uint32_t Difference = 0;
273 uint32_t Distance = 0;
274 uint8_t Is_First_Captured = 0;
275
276 void HCSR04_Read( void )
277 {
278     HAL_GPIO_WritePin(TRIG_GPIO_Port, TRIG_Pin, GPIO_PIN_RESET);
279     delay_us(2);
280
281     HAL_GPIO_WritePin(TRIG_GPIO_Port, TRIG_Pin, GPIO_PIN_SET);
282     delay_us(10);
283     HAL_GPIO_WritePin(TRIG_GPIO_Port, TRIG_Pin, GPIO_PIN_RESET);
284
285     __HAL_TIM_ENABLE_IT(&htim3, TIM_IT_CC4);
286 }
287
288 void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
289 {
290     if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_4)
291     {
292         if(Is_First_Captured == 0)
293         {
294             IC_Val1 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_4);
295             Is_First_Captured = 1;
296             __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_4,
297             TIM_INPUTCHANNELPOLARITY_FALLING);

```

```
297     }
298     else if(Is_First_Captured == 1)
299     {
300         IC_Val2 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_4);
301         __HAL_TIM_SET_COUNTER(htim, 0);
302
303         if(IC_Val2 > IC_Val1) {Difference = IC_Val2 - IC_Val1;}
304         else if(IC_Val1 > IC_Val2) {Difference = IC_Val1 - IC_Val2;}
305
306         Distance = Difference * 34/2000;
307         Is_First_Captured = 0;
308
309         __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_4,
310 TIM_INPUTCHANNELPOLARITY_RISING);
311         __HAL_TIM_DISABLE_IT(&htim3, TIM_IT_CC4);
312     }
313 }
```


APÊNDICE B – LISTA DE MATERIAIS (*BILL OF MATERIALS - BOM*) DO CONJUNTO DIDÁTICO PROPOSTO

Tabela 12 – Lista de componentes para a construção da PCI

Ref.	Componente	Especificações
BZ1	Buzzer	5V - D12mm - H10mm - PTH
C1	100n	Multicamada 50V - PTH
C2	100n	Multicamada 50V - PTH
C3	1uF	Eletrolítico 16-400V - PTH
D1	LED A	5 mm - PTH
D2	LED B	5 mm - PTH
D3	LED ON	5 mm - PTH
D4	LED C	5 mm - PTH
D5	1N4007	PTH
D6	1N4007	PTH
D7	1N4007	PTH
D8	1N4007	PTH
D9	1N4007	PTH
D10	1N4007	PTH
D11	1N4007	PTH
D12	1N4007	PTH
D13	LED LOW	5 mm - PTH
J1	Conn 01x04	Vertical - P2.54mm - PTH
J2	Conn 01x02	Vertical - P2.54mm - PTH
J3	Conn 01x02	Vertical - P2.54mm - PTH
J4	Conn 01x02	Vertical - P2.54mm - PTH
J5	Conn 01x02	Vertical - P2.54mm - PTH
J6	Conn 01x20	Vertical - P2.54mm - PTH
J7	Conn 01x20	Vertical - P2.54mm - PTH
JP1	Jumper	P7.62 mm - PTH
JP2	Jumper	P7.62 mm - PTH
JP3	Jumper	P7.62 mm - PTH
JP4	Jumper	P7.62 mm - PTH
JP5	Jumper	P7.62 mm - PTH
JP6	Jumper	P7.62 mm - PTH
JP7	Jumper	P7.62 mm - PTH
JP8	Jumper	P7.62 mm - PTH
Q1	BD140	TO-126 - PTH

Q2	BD140	TO-126 - PTH
Q3	BD140	TO-126 - PTH
Q4	BD140	TO-126 - PTH
Q5	BC548	TO-92 - PTH
Q6	BC548	TO-92 - PTH
Q7	BC548	TO-92 - PTH
Q8	BC548	TO-92 - PTH
Q9	BD139	TO-92 - PTH
Q10	BD139	TO-126 - PTH
Q11	BD139	TO-126 - PTH
Q12	BD139	TO-126 - PTH
Q13	BC548	TO-92 - PTH
R1	470R	5% - 1/4W - PTH
R2	470R	5% - 1/4W - PTH
R3	470R	5% - 1/4W - PTH
R4	470R	5% - 1/4W - PTH
R5	2,2K	5% - 1/4W - PTH
R6	470R	5% - 1/4W - PTH
R7	470R	5% - 1/4W - PTH
R8	470R	5% - 1/4W - PTH
R9	470R	5% - 1/4W - PTH
R10	150R	5% - 1/4W - PTH
R11	150R	5% - 1/4W - PTH
R12	150R	5% - 1/4W - PTH
R13	150R	5% - 1/4W - PTH
R14	10K	5% - 1/4W - PTH
R15	10K	5% - 1/4W - PTH
R16	30K	5% - 1/4W - PTH
R17	15K	5% - 1/4W - PTH
R18	30K	5% - 1/4W - PTH
R19	15K	5% - 1/4W - PTH
R20	10K	5% - 1/4W - PTH
R21	10K	5% - 1/4W - PTH
R22	10K	5% - 1/4W - PTH
R23	1K	5% - 1/4W - PTH
R24	1K	5% - 1/4W - PTH
R25	1K	5% - 1/4W - PTH
R26	470R	5% - 1/4W - PTH

R27	390R	5% - 1/4W - PTH
R28	10K	5% - 1/4W - PTH
R30	390R	5% - 1/4W - PTH
R31	10K	5% - 1/4W - PTH
RV1	POT	16 mm - PTH
RV2	POT	16 mm - PTH
SW1	SW SPDT	P2.56mm - PTH
SW2	Push Button	6 X 6 X 5mm - 4 terminais - PTH
U1	BluePill STM32F103C	-
U2	BMP180	-
U3	GY-521	-
U4	HC-SR04	Vertical - P2.54mm - PTH
U5	HC-05	-
U6	StepDown CC-CC	MP1584 - PTH
U7	TCRT5000	-
U8	LM324	DIP-14 - W10.16mm - P2.54mm - PTH
U9	TCRT5000	-
U10	LM393	DIP-8 - W7.62mm - P2.54mm - PTH
PCI	PCI	144.4 x 74.5mm - Face simples - 1 oz/sq ft.