PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

MATHEUS HENRIQUE DE CAMPOS MARCOMINI

ANÁLISE DA APLICAÇÃO DE DIFERENTES CONTROLADORES EM ROBÔ SEGUIDOR DE LINHA PARA LEGO® MINDSTORMS® EV3

CAMPINAS

2023

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS
CENTRO DE CIÊNCIAS EXATAS, AMBIENTAIS E DE TECNOLOGIA
PROGRAMA DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

MATHEUS HENRIQUE DE CAMPOS MARCOMINI

ANÁLISE DA APLICAÇÃO DE DIFERENTES CONTROLADORES EM ROBÔ SEGUIDOR DE LINHA PARA LEGO ® MINDSTORMS EV3

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia de Controle e Automação do Centro de Ciências Exatas, Ambientais e de Tecnologia, da Pontifícia Universidade Católica de Campinas, como exigência para obtenção do grau de Bacharel.

Orientador: Professor(a). Dr(a). Lorenzo Campos Coiado

Coorientador: Professor(a). Dr(a). Cecília de Freitas Moraes

CAMPINAS
2023

Ficha catalográfica elaborada por Jerusa Neves dos Santos Lopes CRB 8/10320
Sistema de Bibliotecas e Informação - SBI - PUC-Campinas

629.892 Marcomini, Matheus Henrique de Campos
M321a

Análise da aplicação de diferentes controladores em robô seguidor de linha para LEGO MINDSTORMS EV3 / Matheus Henrique de Campos Marcomini. - Campinas: PUC-Campinas, 2023.

225 f.: il.

Orientador: Lorenzo Campos Coiado.

TCC (Bacharelado em Engenharia de Controle e Automação) - Faculdade de Engenharia de Controle e Automação, Escola Politécnica, Pontifícia Universidade Católica de Campinas, Campinas, 2023.
Inclui bibliografia.

1. Robô. 2. Automação. 3. LEGO. I. Coiado, Lorenzo Campos. II. Pontifícia Universidade Católica de Campinas. Escola Politécnica. Faculdade de Engenharia de Controle e Automação. III. Título.

23. ed. CDD 629.892

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS
CENTRO DE CIÊNCIAS EXATAS, AMBIENTAIS E DE TECNOLOGIA
PROGRAMA DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

MATHEUS HENRIQUE DE CAMPOS MARCOMINI

ANÁLISE DA APLICAÇÃO DE CONTROLE PID EM ROBÔ SEGUIDOR DE LINHA PARA LEGO ® MINDSTORMS EV3

Dissertação defendida e aprovada em [DIA] de [MÊS] e [ANO] pela comissão examinadora:

Prof(a). Dr(a). [NOME]
Orientador e presidente da comissão examinadora.
Pontifícia Universidade Católica de Campinas

Prof.(a). Dr(a). [NOME]
[INSTITUIÇÃO]

Prof.(a). Dr(a). [NOME]
[INSTITUIÇÃO]

CAMPINAS
2023

Dedico esse trabalho à minha família, aos meus amigos, como também aos meus professores.

AGRADECIMENTOS

Nessa seção gostaria de expressar os meus mais sinceros agradecimentos a todos que contribuíram, para a realização deste trabalho de conclusão de curso. Este estudo é a representação mais tangível de todo o empenho dedicado ao curso de Engenharia de Controle e Automação. Ao mesmo tempo também destaca o apoio, a colaboração e participação de inúmeras pessoas não apenas no resultado, mas ao longo de todo o processo.

Dessa forma, em primeiro lugar gostaria de dedicar um parágrafo tanto a minha mãe, Suely Herrera de Campos Pacheco, quanto ao meu pai, Wangner Martins Marcomini, e a minha avó, Philomena Herrera de Campos, como uma forma de homenagem especial, além de merecida. Acredito que muito do que me tornei hoje é devido a gentileza, ao apoio inabalável e ao amor incondicional de cada um deles. Todos tiveram importante participação para consolidar uma orientação sábia, forte, resiliente, como também exemplo inspirador. Eles foram os principais faróis em meu caminho, guiando-me com sabedoria e determinação. O sacrifício e a dedicação de cada um, foi extremamente importante para moldar não apenas este percurso acadêmico, mas também meu caráter e valores.

Também agradeço ao meu tio, a minha tia, aos meus primos e as minhas primas, contudo ressalto principalmente a importância da minha amizade com Lilian Pompeu, Ivanil Pompeu, Daniel Pompeu, Lucy Barchesi, Maria Ligia Pompeu Gentilli Isaac Gentilli, Danilo Pompeu, assim como todos os seus familiares, por terem me acompanhado ao longo dessa caminhada desde o início, e terem me fortalecido pela presença amorosa, além da orientação que me proporcionaram ao longo dos anos. Em conjunto com meus pais vocês foram sem dúvida determinantes no estabelecimento de uma base sólida sobre a qual construí meus sonho e conquistas.

Aos meus professores e as minhas professoras do ensino básico, especialemente Leandro Oliveira, por ter despertado e incentivado meu interesse pela robótica, Ivan Santos, Mariane Schafer e Joares Júnior por terem contribuído imensamente pelo meu interesse pelas ciências exatas, Bianca Teresam e Virgínia de Oliveira pelos seus sábios conselhos.

Também quero expressar minha profunda gratidão ao meu orientador, Lorenzo Campos Coiado, a minha coorientadora, Cecília de Freitas Morais, como também ao professor Everton Dias de Oliveira, cujo auxílio, suporte, orientação sábia e apoio constante de cada um deles foram fundamentais para a conclusão deste trabalho. Suas experiências e insights enriqueceram significativamente a qualidade desta pesquisa.

Aos demais docentes, especialmente aos professores Ralph Robert Heinrich, Franciso de Salles Cintra Gomes, Hamilton da Gama Schroder Filho, Vinicius Gabriel Segala Simionatto, Frank Herman Behrens, Carlos Alberto de Castro Júnior, Eric Alberto de Mello Fagotto e Amilton da Costa Lamas, com quem tive diversas matérias, e pude aprender diferentes conhecimentos importantes para minha trajetória acadêmica e para minha vida.

Aos meus colegas, especialmente Pedro Eiji Tigusa, Matheus Henrique Pereira Martinez, Felipe Gusson Chante, Matheus Augusto de Oliveira, Felipe da Costa Ramos, Matheus da Silva Furlan, Marcelo Mazzochi Filho,Gilberto Paes Lucinda Sobrinho, Rodrigo Araújo Costa, Marcos Vinicios Bortoleti Galli, Vinícius Coelhos Gamas, Gustavo Gonzalez Prates, Emerson de Souza Costa, Bruno de Almeida Miranda, Gabriel Freitas, Amanda Briz de Siqueira, André Ferreira e Silva, Fabian Isabela de Oliveira, Natan Gomes dos Santos e Matheus Oliveira Rodrigues, com os quais convivi a maior parte da graduação, podendo disfrutar das mais diversas situações e compartilhar variadas experiências, além de compartilharem seus conhecimentos, criando um ambiente acadêmico estimulante. A troca de ideias e discussões contribuiu enormemente para o desenvolvimento deste trabalho.

À instituição Pontifícia Universidade Católica de Campinas pelo acesso aos recursos e pela infraestrutura que foram essenciais para a realização das pesquisas e experimentos necessários. Agradeço também aos participantes deste estudo, cuja colaboração foi vital para a coleta de dados e para a validação das conclusões apresentadas.

Por fim, agradeço a todos que, de alguma forma, contribuíram para este projeto, direta ou indiretamente. Este trabalho não teria sido possível sem o apoio generoso de cada um de vocês. Este trabalho é tão de vocês quanto é meu, e cada passo dado foi fortalecido pela presença amorosa e orientação que ambos proporcionaram ao longo dos anos. Obrigado por serem meus heróis, mentores e os melhores pais que alguém poderia desejar. Amo vocês imensamente.

Obrigado por fazerem parte desta jornada.

Atenciosamente,

Matheus Henrique de Campos Marcomini.

“Na jornada do conhecimento, encontramos desafios que moldam nossa compressão, nosso entendimento e nossa intepretação sobre o mundo. Consequentemente, se torna possível adquirir perseverança e tenacidade. Desse modo, assim como um robô pode seguir uma linha traçada, nós podemos enfrentar as complexidades, ajustar nossos parâmetros, e ao final, descobrir que a verdadeira conquista está na jornada, e não necessariamente na linha de chegada”

RESUMO

Este trabalho tem como objetivo principal analisar a complexidade do desenvolvimento de robôs móveis seguidores de linha, mais precisamente a capacidade desses robôs lidarem com a incerteza em seu ambiente, especialmente os erros associados a cada leitura. A seção de Introdução fornece uma visão geral de automação, controle e robótica, destacando momentos, estudos e contribuições essenciais. Nos objetivos, foram destacados os pontos fundamentais para estabelecer um sistema de controle para o robô seguidor de linha autônomo usando o kit LEGO® MINDSTORMS® Ev3. Selecionou-se os principais controladores, incluindo ON/OFF, de 3 Níveis, P, PI, PD e PID, para uma análise detalhada de seu desempenho. Os objetivos específicos envolvem desde a modelagem matemática dos motores e sensores até a implementação, como também avaliação desses controladores em simulações de software, além de hardware. A justificativa destaca a crescente aplicação de robôs autônomos em vários setores, ressaltando a importância deste estudo para uma formação multidisciplinar diante dos desafios da Indústria 4.0. No embasamento teórico, realizou-se uma breve revisão bibliográfica e aprofundamento em conceitos relacionados a robôs seguidores de linha, assim como suas tecnologias. A metodologia descreve as etapas para usar softwares como Matlab e Visual Studio Code, dedicados à simulação virtual e real dos controladores. Em resultados e discussão, desenvolveram-se análises qualitativas e quantitativas para avaliar o desempenho dos controladores em software e hardware. A conclusão baseia-se nos dados obtidos para destacar qual controlador apresentou melhores resultados, abrindo caminho para melhorias futuras na precisão e interação dos robôs com o ambiente. Este trabalho contribui não apenas para a compreensão da engenharia robótica, mas também para áreas como física, matemática e programação, preparando profissionais para os desafios da era da Internet das Coisas.

Palavras-chave: Seguidores de Linha, Kit LEGO® MINDSTORMS® Ev3, Controladores, Modelamento Matemático, Simulação.

ABSTRACT

This work aims to conduct a comprehensive analysis of the challenge in developing mobile line-following robots. The focus is on the ability of these robots to deal with uncertainty in their environment, particularly the errors associated with each reading. The Introduction section provides an overview of automation, control, and robotics, highlighting key moments, studies, and contributions. In the Objectives section, the primary goal is to establish a control system for the autonomous line-following robot built with the LEGO® MINDSTORMS® Ev3 kit. The main controllers, including ON/OFF, 3-Level, P, PI, PD, and PID, were selected for a detailed analysis of their performance. Specific objectives involve mathematical modeling of motors and sensors, as well as the implementation and evaluation of these controllers in software simulations and hardware simulations. The justification emphasizes the growing application of autonomous robots in various sectors, underscoring the importance of this study for multidisciplinary education and preparation for the challenges of Industry 4.0. The Theoretical Framework includes a brief literature review and an in-depth exploration of various concepts related to line-following robots and the employed technologies. The Methodology describes the steps for using software such as Matlab and Visual Studio Code, dedicated to the virtual and real simulation of controller application. In Results and Discussion, qualitative and quantitative analyses were developed to evaluate the performance of controllers in software and hardware. In the Conclusion, the data and information obtained serve as a foundation to highlight which controller yielded better results, paving the way for future improvements in the precision and harmonious interaction of robots with the environment. Therefore, the work not only contributes to the understanding of robotic engineering but also to areas such as physics, mathematics, and programming, offering comprehensive education for professionals facing the challenges of the Internet of Things era.

Keywords: Line followers, LEGO® MINDSTORMS® Ev3 Kit, Controlers, Mathematic Modeling, Simulation.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE EQUAÇÕES

LISTA DE SIGLAS OU ABREVIATURAS

ARM	Nome de uma companhia Britânica produtora de chips.
CPU	Central Processing Unit – Unidade de Processamento Central
EEPROM	Eletrically Erasable Programmable Read-Only-Memory – Memória Somente-Leitura Programável Eletricamente Apagável

LISTA DE SÍMBOLOS

A	Unidade de medida de corrente elétrica – Ampère(s)
b	Unidade de medida de armazenamento – bits
B	Unidade de medida de armazenamento – Byte
Kb	Unidade de medida de armazenamento – Kilobit(s)
KB	Unidade de medida de armazenamento – Kilobyte(s)
Mb	Unidade de medida de armazenamento – Megabit(s)
MB	Unidade de medida de armazenamento – Megabyte(s)
H	Unidade de medida de indutância elétrica – Henry(s)
Hz	Unidade de medida de frequência – Hert(s)
Kg	Unida de medida de massa – Quilograma(s)
m	Unidade de medida de espaço – Metro(s)
N	Unidade de medida de força – Newton(s)
Rad	Unidade de medida de ângulo – Radiano(s)
s	Unidade de medida de tempo – Segundo(s)
ms	Unidade de medida de tempos – Milissegundo(s)
Ω	Unidade de medida de resistência elétrica – Ohm(s)
V	Unidade de medida de tensão elétrica – Volt(s)

SUMÁRIO

1. INTRODUÇÃO:

1.1 CONTEXTO HISTÓRICO:

A trajetória da humanidade é marcada por uma evolução contínua, que se estende desde a Pré-História até a Idade Contemporânea. Ao longo da história, os seres humanos têm enfrentado uma grande quantidade de mudanças em seu cotidiano. Como resultado, revoluções têm consistentemente marcado o início de novos ciclos, e a conclusão de ciclos antigos. Essa dinâmica ilustra que os aspectos econômicos, políticos e sociais permanecem sujeitos a uma transformação contínua (BRYNJOLFSSON; MACFEE, 2014; SCHWAB, 2016).

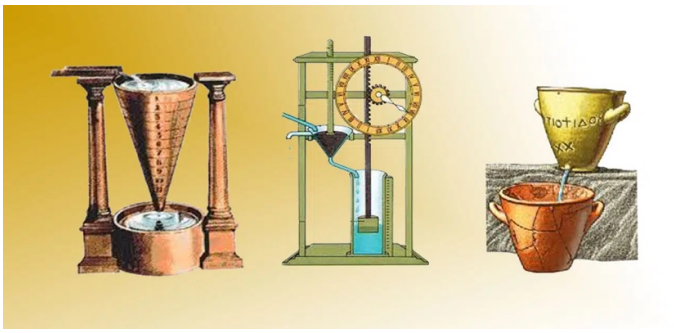
A observação também se aplica ao campo tecnológico. Ao longo dos anos, a tecnologia tem evoluído em conjunto com as sociedades, desencadeando constante transformação. Desde a transição do nomadismo para o sedentarismo, quando os primeiros hominídeos dominaram o fogo, novos conhecimentos e técnicas têm surgido e sido reinventados conforme as eras (BRYNJOLFSSON; MACFEE, 2014; SCHWAB, 2016).

Essas inovações possibilitaram a transformação de pequenas comunidades em grandes centros urbanos e de modestas plantações em extensas propriedades rurais, resultando na expansão da comunicação, produção e transporte para níveis de globalização inimagináveis até então (BRYNJOLFSSON; MACFEE, 2014; SCHWAB, 2016).

Após a conclusão da Primeira Revolução Industrial no século XVIII, seguida pela Segunda Revolução Industrial no século XIX e pela Terceira Revolução Industrial no século XX, a sociedade contemporânea está agora testemunhando o surgimento de uma nova era revolucionária. Essa nova fase é impulsionada pela ascensão da Indústria 4.0, da Internet das Coisas (IoT), da Inteligência Artificial e da análise de grandes volumes de dados (Big Data) (BENNET, 1993; NISE, 2013; RIBEIRO, 2001).

Os sistemas de controle têm uma origem profundamente enraizada na automação. Ao longo dos séculos, podemos encontrar os antecessores desses conceitos nos gregos, que foram pioneiros no desenvolvimento de relógios de água (Figura 1). Embora haja variações de interpretação, é inegável a influência dessas primeiras inovações no campo da automação. (BENNET, 1993; NISE, 2013).

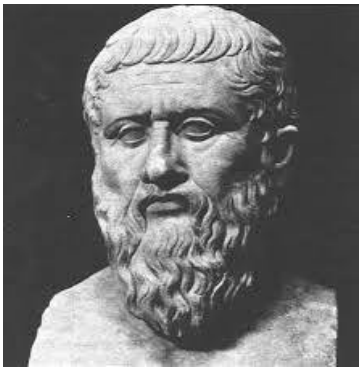
Figura 1 - Relógio de Água (Clepsidra).



Fonte: Segredos do Mundo (2023).

A história dos sistemas de controle retroalimentados remonta a cerca de 300 a.C., quando Ctesíbios (Figura 2), um matemático e engenheiro grego, se destacou como um dos primeiros pioneiros ao inventar um relógio de água chamado Clepsydra. Esse dispositivo exigia um nível constante no reservatório de abastecimento. Para resolver esse desafio, Ctesíbios utilizou uma válvula de flutuação semelhante às encontradas nos vasos sanitários modernos, a fim de realizar esse controle preciso (BENNET, 1993; NISE, 2013).

Figura 2 – Ctesíbios.



Fonte: Computer Timeline.

Mais tarde, o princípio de controle líquido foi aplicado por Filão de Bizâncio (Figura 3), um autor, engenheiro e inventor grego. Ele utilizou o conceito de controle de nível líquido em uma lâmpada a óleo, com o objetivo de manter a emissão de luz constante com base na queima do combustível. Isso foi conseguido por meio de um mecanismo vertical, que permitia a manutenção da iluminação assim que o fluido estivesse disponível nos reservatórios (BENNET, 1993; NISE, 2013).

Figura 3 - Filão de Bizâncio.

Fonte: Philosophy Basics (2023).

Durante a Antiguidade, as civilizações antigas já incluíam em suas culturas elementos que sugeriam a criação de entidades autônomas capazes de agir independentemente de seus criadores. Essas inovações transcendiam a vontade dos inventores e podem ser observadas nos registros das primeiras sociedades humanas, como apontado por vários estudiosos. Além disso, o conceito de robótica também tem suas raízes nessa época (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

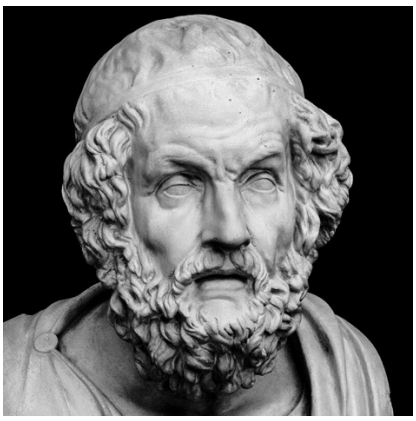
Durante esse período, os gregos associaram essas inovações à figura mitológica do deus Hefesto (Figura 4), cuja imagem estava intimamente ligada ao artesanato e à metalurgia, destacando-se por sua relação com a produção tecnológica (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

No entanto, foi na obra "Ilíada" de Homero (Figura 5), datada do século VIII a.C., que Hefesto foi imortalizado, retratado como um habilidoso criador de máquinas que possuíam semelhanças com seres humanos (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Figura 4 – Estátua do Deus Grego Hefesto do escultor Guillaume Cousto.

Fonte: Toda Matéria (2023).

Figura 5 – Escultura do Busto de Homero, autor desconhecido.

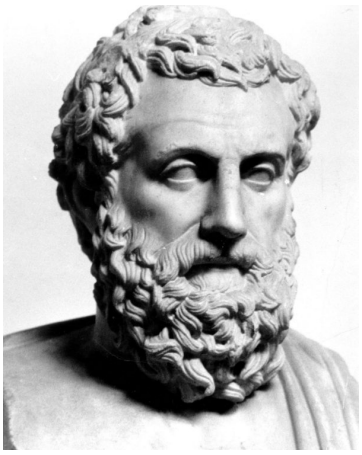


Fonte: Poetria – Poesia e Teatro (2023).

Em diferentes contextos, essas máquinas muitas vezes incorporavam características animais e frequentemente assumiam a forma de sistemas compostos por pesos e bombas pneumáticas. Esses componentes desempenhavam um papel crucial ao conferir independência de movimento a essas criações (NIKU, 2001).

No entanto, a robótica não se limitava apenas ao domínio mitológico. Na filosofia, estudiosos já exploravam os conceitos essenciais da robótica. Como mencionado por Aristóteles (Figura 6) no século IV a.C. (322 a.C.), "Se os instrumentos pudessem realizar suas próprias tarefas, obedecendo ou antecipando a vontade das pessoas..." (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; SIEGWART; NOURBAKHSH, 2004).

Figura 6 – Busto do filósofo grego Aristóteles.

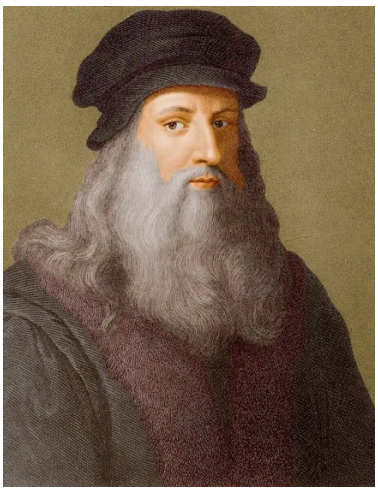


Fonte: Hypescience (2016).

Outros inventores também contribuíram de forma significativa para o avanço da robótica ao longo do tempo. Um exemplo notável é o de Leonardo da Vinci (Figura 7), em 1495, quando concebeu um robô antropomórfico, ou seja, com semelhanças humanas (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; SIEGWART; NOURBAKHSH, 2004).

Esse autômato era uma representação de um cavaleiro medieval, vestido com armadura e empunhando uma espada. Além disso, ele possuía a habilidade de se mover de maneira semelhante a um ser humano (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; SIEGWART; NOURBAKHSH, 2004).

Figura 7 – Gravura de Leonardo Da Vinci.



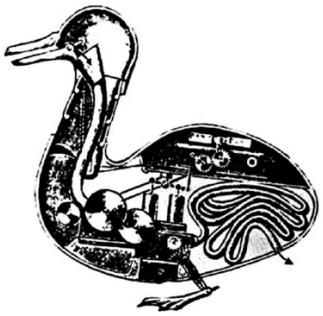
Fonte: Universo Racionalista (2021).

Em 1738, Jacques de Vaucanson (Figura 8) alcançou um marco importante ao desenvolver uma série de autômatos. Um deles, em particular, era um pato (Figura 9) com habilidades impressionantes, que podia emitir sons, mover suas asas e até mesmo simular ações como comer e digerir alimentos (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Figura 8 - Jacques de Vaucanson.

Fonte: Advancing Physics (2018).

Figura 9 – Pato Autômato.



Fonte: ResearchGate (2011).

Em um estágio seguinte, os sistemas de controle realimentados deram um salto tecnológico no século XVIII, graças a Edmund Lee e William Cubbit, os quais conceberam um sistema de controle de velocidade para moinhos de vento (Figura 10) (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

Com isso, fica claro que a produção de alimentos, especialmente grãos e cereais, começou a adquirir características mais automatizadas, uma vez que sistemas autônomos estavam gradualmente substituindo a mão de obra humana em diversas tarefas (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

No entanto, essa automação ainda se limitava a certas etapas da produção, uma vez que a maioria das atividades ainda dependia do trabalho humano (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

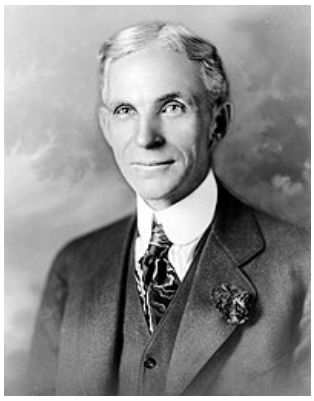
Figura 10 - Moinho de Vento.

Fonte: Conexão Amsterdam (2020).

Após algum tempo, essa dinâmica passou por mudanças significativas durante a Primeira Revolução Industrial. A invenção dos motores a vapor, juntamente com o regulador de velocidade desenvolvido por James Watt, desempenhou um papel fundamental nessa transformação (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

A automação começou a se tornar uma parte essencial da indústria, impulsionada pelas contribuições notáveis de Henry Ford (Figura 11) e Frederick Taylor (Figura 12). Eles introduziram conceitos como divisão do trabalho, delegação de tarefas, integração de máquinas nas fábricas, formação de estoques e busca incessante pela máxima produtividade (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

Figura 11 – Henry Ford.



Fonte: SóCientífica (2022).

Figura 12 – Frederick Taylor.

Fonte: Edições Sílabo (2023).

No entanto, essas inovações representaram apenas os primeiros passos em direção à automação e aos sistemas de controle. Embora tenham tido um impacto significativo nas sociedades da época e, consequentemente, na história, elas não se mostraram completamente determinantes na definição do que viria a ser reconhecido como Automação Industrial e a Teoria de Sistemas de Controle (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

Nesse sentido, começou a surgir uma distinção sutil entre os significados das duas áreas, permitindo uma diferenciação mais objetiva. Foi somente no século XIX que ocorreram as primeiras mudanças realmente significativas nesse campo do conhecimento (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

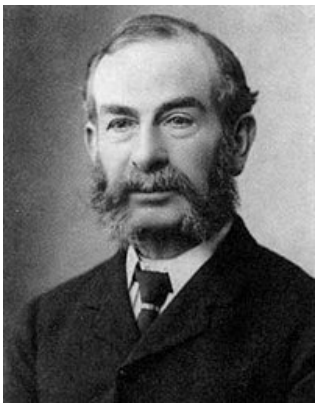
Isso se deu com a criação do critério de estabilidade por James Clerk Maxwell (Figura 13) e sua posterior expansão por Edward John Routh (Figura 14), que resultou no desenvolvimento do critério de estabilidade conhecido como critério de Routh-Hurwitz (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

Mais adiante, essa abordagem foi estendida ao estudo de sistemas não lineares por Alexander Michailovich Lyapunov (Figura 15), por meio do critério de Lyapunov (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

Figura 13 – James Clerk Maxell.

Fonte: Britannica (2023).

Figura 14 – Edward John Routh.



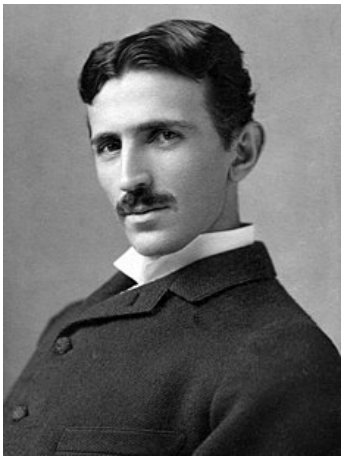
Fonte: MacTutor (2023).

Figura 15 – Alexander Michailovich Lyapunov.

Fonte: Towards Data Science (2021).

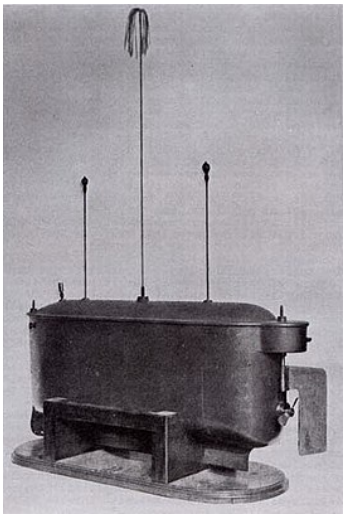
Simultaneamente, em 1898, Nikola Tesla (Figura 16) fez uma excelente contribuição para a robótica ao desenvolver um barco controlado remotamente (Figura 17). Vale destacar suas palavras: "I treated the whole field broadly, not limiting myself to mechanics controlled from a distance, but to machines possessed of their own intelligence. Since that time had advanced greatly in the evolution of the invention and think that the time is not distant when I shall show an automation which left to itself, will act as though possessed of reason and without any willful control from the outside" (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Figura 16 – Nikola Tesla.



Fonte: Museu WEG (2015).

Figura 17 – Barco de Controle Remoto.



Fonte: Hyper Cultura (2017).

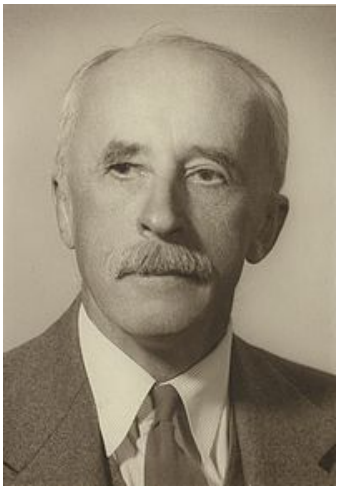
No século XIX, houve um significativo avanço no desenvolvimento de sistemas de controle para automação de direção, condução e estabilização. Diversos nomes contribuíram para esse progresso, como Henry Bessemer (Figura 18), Nicholas Minorsky (Figura 19) e a Sperry Gyroscope Company, que introduziu inovações no contexto naval (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

Os estudos de Minorsky, em particular, desempenharam um papel importante ao estabelecer as bases para a criação dos controladores proporcionais, integrais e derivativos (PID), que são amplamente utilizados atualmente (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

Figura 18 – Henry Bessemer.

Fonte: National Inventors Hall of Fame (2023).

Figura 19 – Nicholas Minorsky.



Fonte: Wikipedia (2023).

No século XX, figuras como H.W. Bode (Figura 20), H. Nyquist (Figura 21) e Walter R. Evans (Figura 22) fizeram contribuições significativas. Durante as décadas de 1920 e 1930, H.W. Bode e H. Nyquist permitiram a análise de sistemas no domínio da frequência (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

Já no final da década de 1940, Walter R. Evans desenvolveu uma abordagem matemática que representava as raízes de uma equação relacionada ao comportamento de sistemas de realimentação (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

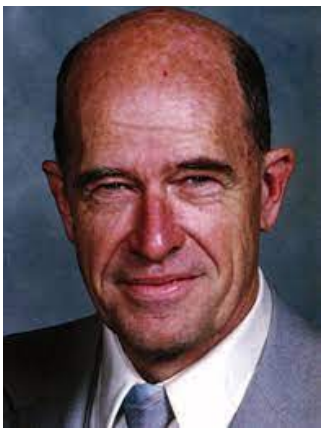
Figura 20 – H.W. Bode.

Fonte: National Academy of Engineering (2023).

Figura 21 – H. Nyquist.

Fonte: Sematic Scholar (2011).

Figura 22 – Walter R. Evans.



Fonte: IEEE Xplorer (2004).

Durante esse mesmo período, várias definições e termos relacionados à robótica foram formulados, inicialmente encontrando expressão na literatura de ficção científica. Um exemplo é o termo "robô", que foi apresentado pelo escritor tcheco Karel Čapek em sua obra "R.U.R." (Rossum's Universal Robots) em 1921 (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Essa palavra tem suas origens no vocábulo "robota", que se referia a formas de trabalho forçado e compulsório. Com isso, Čapek estabeleceu os robôs como criações resultantes da engenharia biológica, projetados para atender às necessidades humanas (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Figura 23 – Karel Capek.

Fonte: Britannica (2023).

Em um estágio posterior, nos anos 1926 e 1927, ocorreu outro momento crucial para o desenvolvimento do conceito de robô por meio do filme "Metrópolis" de Fritz Lang. Assim como Karel Čapek, Lang também utilizou o conceito de robô como um meio de transmitir uma crítica social ao trabalho forçado e opressivo (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Figura 24 – Fritz Lang.

Fonte: Cine Players (2023).

As contribuições de Isaac Asimov durante a década de 1940 ampliaram o conceito, atribuindo o termo "Robótica" para nomear uma disciplina científica dedicada ao estudo da construção de robôs (ASIMOV, 1969).

A narrativa "Runaround" foi responsável por introduzir esse termo, enquanto a compilação "Eu, Robô" consolidou as concepções de Asimov. Nessa obra, foram estabelecidas as renomadas "Três Leis da Robótica" da seguinte maneira (ASIMOV, 1969):

1ª Lei: “Um robô não pode ferir um ser humano ou, permanecendo passivo, deixar um ser humano exposto ao perigo” (ASIMOV, 1969).

2ª Lei: “O robô deve obedecer às ordens dadas pelos seres humanos, exceto se tais ordens estiverem em contradição com a primeira lei” (ASIMOV, 1969).

3ª Lei: “Um robô deve proteger sua existência na medida em que essa proteção não estiver em contradição com a primeira e a segunda leis” (ASIMOV, 1969).

Mais tarde, foi desenvolvida a 4ª Lei, apelidada de Lei 0, estabelecendo que: “Um robô, não pode causar mal à humanidade ou, por omissão, permitir que a humanidade sofra algum mal” (ASIMOV, 1969).

Figura 25 – Isaac Asimov.

Fonte: Universidade Federal de Minas Gerais (2023).

A automação começou a ser progressivamente incorporada nos setores automobilístico e químico. Tal incorporação foi impulsionada pela necessidade de reforçar a segurança nas instalações fabris. Nesse contexto, uma substituição cada vez mais intensa da força de trabalho humana por robôs ocorreu, particularmente em tarefas repetitivas e ambientes perigosos (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

A robótica encontrou terreno fértil no setor industrial ao longo das décadas de 1950 e 1960. Esses períodos testemunharam avanços tecnológicos mais substanciais nas áreas de automação e controle, com a introdução de sensores mais avançados e atuadores mais robustos. Assim, uma significativa progressão foi alcançada na capacidade de movimento autônomo dos robôs (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

No início da década de 1950, William Grey Walter alcançou um feito importante ao desenvolver tartarugas robóticas, que já demonstravam uma perceptível capacidade de compreender o ambiente circundante (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Mais adiante, em 1954, George Devol cunhou o termo "Universal Automation", inaugurando as primeiras incursões no campo dos robôs programáveis com a criação de robôs transportadores. Enquanto isso, Joe Engelberger recebeu o título de "o pai da robótica", ao concretizar a criação do primeiro robô comercial pela empresa Unimation, denominado UNIMATE (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

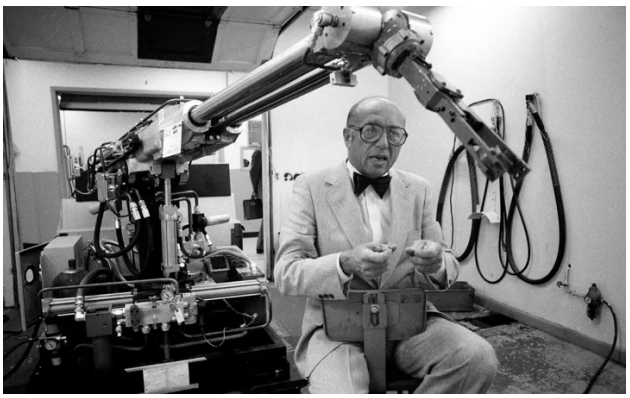
Figura 26 – William Grey Walter.

Fonte: Science Direct (2005).

Figura 27 – George Devol.

Fonte: Computer Human Experience (2023).

Figura 28 – Joseph Engelber.



Fonte: The New York Times (2015).

Assim, gigantes industriais como a General Motors embarcam na adoção dessas tecnologias em suas linhas de produção. Concomitantemente, ao vislumbrar uma notável oportunidade em um campo relativamente novo, o Stanford Research Institute concebe o primeiro robô móvel com capacidades excepcionais de movimentação tarefas (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWAT; NOURBAKHSH, 2004).

Batizado de Shakey, esse robô incorporava sensores em sua estrutura, permitindo-lhe traçar uma espécie de raciocínio a cada passo tomado durante a execução de tarefas (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Com a introdução dos computadores, tanto a disciplina da automação quanto a de controle alcançaram um patamar tecnológico inédito. As inovações transcenderam sua natureza meramente física (tangível), incorporando uma dimensão virtual (intangível) (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

Isso viabilizou a Teoria de Controle Moderna não apenas para resolver desafios típicos enfrentados no passado, mas também para abordar novos obstáculos que demandavam a formulação de soluções inéditas (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

Durante as décadas de 1960 e 1970, a introdução de microprocessadores e sistemas de controle digital promoveu a automação completa de processos industriais. Essa evolução viabilizou a criação de sistemas de produção automatizados e flexíveis, capazes de se ajustar às diferentes demandas de fabricação, ao mesmo tempo em que se testemunhava a crescente inserção de robôs no âmbito industrial (ÂNSTRÖM; KUMAR, 2014; BENNET, 1993; BRYNJOLFSSON; MACFEE, 2014; NISE, 2013; SCHWAB, 2016).

Entre a década de 1970 e a década de 1980, houve um avanço no desenvolvimento de braços mecânicos ou manipuladores robóticos. Empresas, institutos e universidades contribuíram significativamente para a robótica, tanto no setor industrial como militar (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Assim, robôs como o Unimate PUMA e SCARA desempenharam um papel fundamental ao impulsionar ainda mais os progressos nesse campo. Porém, dentre essas contribuições, um marco de extrema relevância para o estudo a ser conduzido é a concepção do primeiro robô seguidor de linha pela Stanford University. Esse evento desempenhou um papel crucial na difusão global desse tipo de robótica (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Na década de 1990 e no início do século XXI, a automação em conjunto com sistemas de controle transcendeu o âmbito industrial, estendendo-se para outras esferas como agricultura, medicina e logística (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Paralelamente, a robótica vivenciou um incrível progresso nesse período, dando origem a robôs cada vez mais avançados, dotados da capacidade de executar tarefas previamente reservadas exclusivamente aos seres humanos (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Nesse mesmo período, na esfera educacional, surgiram os primeiros kits de robótica, sendo a empresa LEGO® uma das pioneiras com o lançamento da primeira versão do MINDSTORMS em 1998 (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

1.2 PROBLEMATIZAÇÃO:

De acordo com o pensamento de LOZANO-PÉREZ (1990), o maior obstáculo para o desenvolvimento de um robô móvel é sua capacidade de trabalhar com a incerteza presente no ambiente ao seu redor.

Sendo assim, é necessário que ele possa apresentar um nível de características de interpretação sensorial, que possibilite ao robô se localizar no ambiente, como também visualizar possíveis objetos (obstáculos) que o cercam.

Ao mesmo tempo, ele também deve apresentar um nível de características de raciocínio, que possibilite ao robô compreender e entender quais movimentos devem ser realizados para a condução de uma dada tarefa.

1.3 OBJETIVO GERAL:

Esse trabalho tem como principal objetivo assegurar o desenvolvimento de um sistema de controle relacionado a um robô autônomo móvel seguidor de linha, construído a partir do kit de robótica LEGO® MINDSTORMS® Ev3, e elaborar um código responsável por estabelecer a criação de um controlador, com base no uso de softwares dedicados.

Adicionalmente, pretende-se avaliar os diferentes desempenhos dos controladores ON/OFF, de 3 Níveis, P (Proporcional), PI (Proporcional-Integral), PD (Proporcional-Derivativo) e PID (Proporcional-Integral-Derivativo) empregados no robô autônomo seguidor de linha dentro de uma trajetória previamente estabelecida, ao executar um dado conjunto de tarefas.

1.4 OBJETIVOS ESPECÍFICOS:

Modelar matematicamente a função de transferência do robô autônomo seguidor de linha:

- Realizar o modelamento matemático dos parâmetros cinemáticos associados aos motores empregados na montagem do robô. (Motores grandes e motor médio);
- Realizar o modelamento matemático dos parâmetros elétricos associados aos motores empregados na montagem do robô. (Motores grandes e motor médio);
- Realizar o modelamento matemático do funcionamento dos sensores empregados na montagem do robô. (Sensor de luz e sensor infravermelho).

Modelar trajetórias e movimentação suave:

- Planejar e construir um circuito de simulação a ser percorrido pelo robô;
- Planejar o movimento das rodas seguindo um padrão simétrico e suave.

Estudar e aprofundar a Teoria de Controle Digital:

- Analisar a aplicação do controle em malha aberta (sem realimentação);
- Analisar a aplicação do controle em malha fechada (com realimentação);
- Investigar a aplicação de controladores PID e suas respectivas variações, ou seja, controlador P (proporcional), controlador PI (Proporcional-Integral), PD (Proporcional-Derivativo) e controladores mais simples, como, por exemplo o ON/OFF e o de 3 Níveis.

Realizar simulações (Software):

- Superar as limitações associadas a programação em blocos do software educacional fornecido pela LEGO®, utilizando a plataforma Visual Studio Code em conjunto da extensão MicroPython e ev3dev para o desenvolvimento de um sistema de aquisição de dados referente ao comportamento dos componentes presentes na construção do robô, em linguagem de programação Python.
- Implementação da função de transferência do robô e dos controladores desenvolvidos por meio da utilização do software MATLAB, juntamente com a construção de Diagramas de Blocos, utilizando o recurso SIMULINK, em linguagem de Programação M.

Realizar simulações (Hardware):

- Avaliar o desempenho dos controladores construídos, desde o mais simples ON/OFF, até os mais complexos PID (Proporcional-Integral-Derivativo), e suas respectivas combinações, P (Proporcional), PI (Proporcional-Integral) PD (Proporcional-Derivativo) com o robô seguidor de linha.

1.5 JUSTIFICATIVAS:

A robótica é um campo de conhecimento em contínuo crescimento, evolução e expansão. Ao longo dos anos, sua incorporação em novas áreas tem sido crescente, abrangendo desde as ciências exatas até as biológicas e humanas, encontrando aplicações nos setores de educação, indústria e saúde. Dessa forma, novas tecnologias têm se integrado ao cotidiano humano, realizando desde tarefas simples até as mais complexas.

Os robôs autônomos seguidores de linha são um exemplo concreto dessa evolução, pois possuem inerentemente a habilidade de movimentar-se ao longo de trajetos definidos, orientando-se por linhas previamente traçadas. Inicialmente, seu papel estava associado à indústria, contribuindo em atividades como produção e logística. Assim, suas atribuições frequentemente estão ligadas aos centros de distribuição, seja para despachar mercadorias ou para auxiliar no transporte de materiais e peças nas instalações fabris.

No entanto, suas aplicações podem ser estendidas a outros campos, como a saúde. Em ambientes hospitalares, por exemplo, eles podem desempenhar um papel crucial ao distribuir medicamentos nas diversas alas conforme a demanda de cada paciente, operando sob a supervisão de médicos e enfermeiros. Esse avanço na automação pode otimizar o gerenciamento dos suprimentos médicos, permitindo um atendimento mais eficaz e centrado nas necessidades individuais de cada paciente.

Considerando a inserção dessas inovações nos principais ambientes urbanos, torna-se claro que a materialização do conceito de cidades inteligentes está mais tangível do que nunca. Além disso, uma aplicação adicional reside na entrega de mercadorias e alimentos por meio de vias dedicadas, como evidenciado por alguns testes preliminares envolvendo drones.

Na esfera agrícola, esses robôs poderiam desempenhar funções como pulverização e irrigação entre as fileiras de plantas, além de contribuírem nas tarefas de colheita. No domínio da pecuária, poderiam abastecer regularmente as baias com ração para os animais. No que tange à segurança, sua utilidade se traduziria em monitorar e supervisionar ambientes, conduzindo patrulhas de acordo com trajetos pré-definidos.

No contexto da educação primária, esses robôs poderiam ser empregados como ferramentas introdutórias para alunos, abordando conceitos relacionados à eletrônica e à programação. Em níveis de ensino mais avançados, a análise do comportamento e do desempenho dos robôs, utilizando controladores como o PID, poderia contribuir para o desenvolvimento de conhecimentos na área de controle e automação.

A utilização de kits de robótica baseados em microcontroladores, como o LEGO® MINDSTORMS® Ev3, está se tornando cada vez mais comum em escolas e universidades, solidificando-se como uma ferramenta de aprendizado. Sua facilidade de manuseio contribui para o desenvolvimento de projetos tecnicamente mais viáveis.

A análise do comportamento de um controlador PID aplicado a um robô autônomo seguidor de linha tem o poder de aprofundar o entendimento não apenas na área da engenharia, mas também em campos como a física, a matemática e a programação.

Isso abrange desde a criação de modelos e algoritmos até a elaboração de códigos, a construção de protótipos e a verificação de resultados por meio de simulações em hardware e software.

Contribuindo, por sua vez, para a consolidação de uma formação multidisciplinar do profissional, capacitando-o para diversas esferas do mercado de trabalho. Sendo especialmente relevante com o advento da Indústria 4.0 e a implementação da Internet das Coisas (IoT – Internet of Things).

Consequentemente, surge a necessidade de conduzir um estudo aprofundado sobre a aplicação de controladores voltados para essa categoria específica de robôs autônomos. O objetivo primordial é garantir não apenas uma precisão máxima em seus movimentos, como também uma capacidade contínua de interação harmoniosa com o ambiente.

Englobando habilidades como a evasão de obstáculos, promovendo tanto produtividade quanto segurança. Simultaneamente, esse estudo visa estender os resultados obtidos para a compreensão de uma ampla gama de trajetórias, contemplando uma variedade diversificada de situações.

2. EMBASAMENTO TEÓRICO:

A construção do embasamento teórico foi dividida em duas seções. A primeira seção apresenta a fundamentação teórica de alguns conceitos técnicos básicos, com o objetivo de estabelecer algumas convenções que serão utilizadas ao longo da análise. Dessa forma, as referências empregadas são em grande parte livros e artigos-científicos tratando sobre automação, sistemas de controle e robôs autônomos veiculares.

Enquanto a segunda seção apresenta a fundamentação teórica voltada a metodologia de tecnologias capazes de serem empregadas para a resolução do problema central em análise, com o objetivo de demonstrar a capacidade técnica delas. Dessa forma, as referências empregadas são em grande parte artigos tratando sobre robôs autônomos veiculares construídos utilizando kits de robótica.

2.1 Definição: Automação

Como enfatizado na introdução deste trabalho, a robótica solidificou sua posição como uma disciplina intrinsecamente ligada ao cenário industrial. A empregabilidade dos braços robóticos acabou ganhando bastante espaço na linha de produção ao demonstrar a capacidade de substituir eficientemente a força de trabalho humana (LOZANO-PÉREZ, 1990).

No entanto, essa definição apenas esboça a amplitude dessa disciplina, pois a enxerga como um mero desdobramento da automação. Ao longo dos anos, a automação, os sistemas de controle e a robótica evoluíram em paralelo e interligados. No entanto, é crucial entender que essa interdisciplinaridade não implica em igualdade de significados entre esses três termos (LOZANO-PÉREZ, 1990).

A automação é essencialmente fundamentada na substituição da força de trabalho humana pela operação mecanizada. Contudo, não se deve confundir essa abordagem com a simples mecanização, que envolve a utilização de máquinas controladas por seres humanos em linhas de produção ou montagem (RIBEIRO, 2001).

A automação é a realização de uma atividade, de um processo, ou de uma tarefa com interferência mínima do ser humano, ou até mesmo sem a interferência dele, possibilitando a ocorrência de acordo com uma ação intrínseca ao maquinário (RIBEIRO, 2001).

Entretanto, é necessário ressaltar que essa ação está submetida a um intervalo de tempo específico, como também a uma resposta desejada, isto é, condições de controle do sistema (RIBEIRO, 2001).

Consequentemente, outras facetas da disciplina frequentemente são negligenciadas. A construção e programação de robôs autônomos, seja na forma de animais, humanoides, ou até mesmo veículos, perdem espaço, pois inicialmente não apresentam uma aplicação simples na linha de produção como os manipuladores mecânicos (BENNET, 1993; BOLTON, 1995; CRAIG, 2005; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; LOZANO-PÉREZ, 1990; NISE, 2013; OGATA, 2014).

2.2 Definição: sistemas de controle

Dessa maneira, a ausência de robôs autônomos na linha de produção é decorrente da necessidade de implantação de sistemas de controle mais robustos, além de um estudo mais profundo sobre a interação dos componentes ou dos dispositivos hidráulicos, elétricos, mecânicos e pneumáticos presentes na composição dos robôs (BENNET, 1993; BOLTON, 1995; CRAIG, 2005; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; LOZANO-PÉREZ, 1990; NISE, 2013; OGATA, 2014).

Paralelamente, exigindo a construção de modelos matemáticos, capazes de garantir uma representação adequada dos fenômenos biológicos, físicos, químicos atuantes durante o funcionamento, devido a presença de incertezas e ruídos (BENNET, 1993; BOLTON, 1995; CRAIG, 2005; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; LOZANO-PÉREZ, 1990; NISE, 2013; OGATA, 2014).

Existe uma grande dificuldade em realizar estudos considerando a aplicação da robótica para ambientes desconhecidos, isto é, onde eventos não presentes na programação dos robôs podem acontecer com uma certa frequência (BENNET, 1993; BOLTON, 1995; CRAIG, 2005; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; LOZANO-PÉREZ, 1990; NISE, 2013; OGATA, 2014).

Sendo assim, o esforço empregado para o desenvolvimento de estudos nesse campo é considerado extremamente custoso e amplamente cansativo. Portanto, o ambiente industrial tem uma preferência pela utilização de modalidades de robôs fixos, apresentando uma liberdade mais limitada dentro do chão de fábrica (BENNET, 1993; BOLTON, 1995; CRAIG, 2005; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; LOZANO-PÉREZ, 1990; NISE, 2013; OGATA, 2014).

Essa visão de trabalhar com o desconhecido está extremamente relacionada ao papel de um sistema de controle. O conceito de sistema de controle, inicialmente se estabelece na forma de uma ideia bastante abstrata, pois o modo mais simples de compreender, é visualizá-lo como uma caixa preta, e suas características principais sendo a presença de uma entrada e de uma saída (BENNET, 1993; BOLTON, 1995; CRAIG, 2005; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; LOZANO-PÉREZ, 1990; NISE, 2013; OGATA, 2014).

Figura 29 – Simplificação de um sistema dinâmico.

Fonte: Adaptado de Nise (2013-p.72).

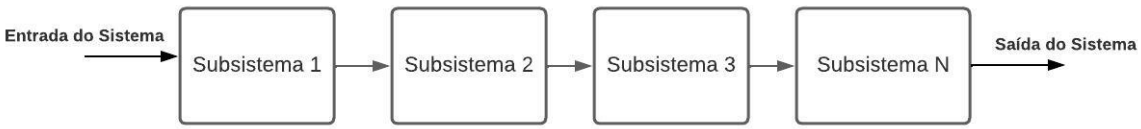
Essa simplificação estabelece um desconhecimento em relação ao conteúdo da caixa, mas é possível fornecer um sinal de entrada como referência, fazendo a saída assumir um dado valor (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

A caixa preta, ou seja, o sistema, é capaz de apresentar uma estrutura muito simples, ou muito complexa, possuindo componentes e elementos interagindo entre si, e até mesmo configurando relações extremamente típicas em cada uma das seções, estruturando subsistemas. (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

Em certas situações, é mais vantajoso realizar os estudos sobre cada um dos subsistemas, imaginando que eles são caixas pretas menores, que compõe o sistema como um todo. Dessa forma, sabendo os dados e informações de cada uma das entradas, como também de cada uma das saídas dos subsistemas, a relação do sistema propriamente dito é passível de obtenção (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

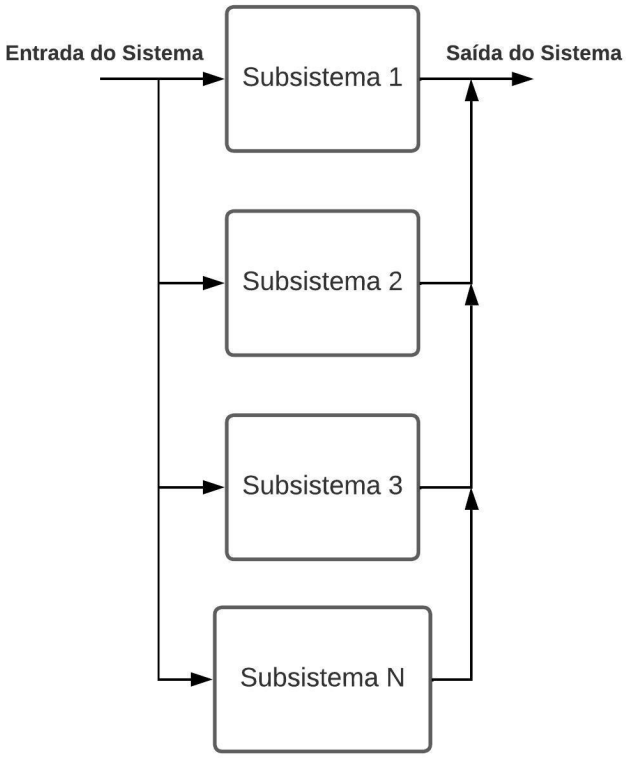
A Figura 30 apresenta um sistema composto por um conjunto de subsistemas associados em série, enquanto a Figura 31 apresenta outro sistema composto por um conjunto de subsistemas associados em paralelo.

Figura 30 – Subsistemas associados em série.



Fonte: Adaptado de Nise (2013-p72).

Figura 31 – Subsistemas associados em paralelo.



Fonte: Adaptado de Nise (2013-p72).

Mesmo existindo uma grande variedade de sistemas, em muitas situações os seus respectivos funcionamentos apresentam semelhanças, a ponto de um estudo estabelecido sobre uma dada entrada e uma dada saída, possibilitar avaliar outros diferentes de modo a obter um mesmo comportamento (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

Segundo NISE (2013), existem outras vantagens para a condução de um estudo de acordo essa linha de interpretação, sendo elas:

- Amplificação: Um sistema de controle pode possuir a capacidade de fornecer um ganho ao sinal de entrada fazendo com que o sinal de saída tenha um valor maior, e até mesmo um valor menor, dependendo do caso. Ou seja, uma amplificação de sua potência, ou atenuação de sua potência (NISE, 2013).
- Controle remoto: Um sistema de controle pode possuir a capacidade de ser controlado remotamente a curta distância e a longa distância para realizar tarefas em ambientes inacessíveis aos seres humanos (NISE, 2013).
- Conveniência na forma de entrada: Um sistema de controle pode possuir a capacidade de modificar o valor assumido por alguma grandeza biológica, física ou química presente no ambiente. Desse modo, assegurando as características desejadas (NISE, 2013).
- Compensação de perturbações: Um sistema de controle pode possuir a capacidade de compensar a influência de uma grandeza biológica, física ou química indesejada atuante sobre o sinal de entrada empregado. Dessa forma, consegue amenizar os efeitos possíveis sobre a saída do sistema (NISE, 2013).

Portanto, combinando o ideal de automação e o ideal de sistema de controle é possível impulsionar o desenvolvimento da robótica, tanto no campo industrial (atuação mais específica), quanto no campo científico (atuação mais abrangente), uma vez que a precisão é exigida em ambas, mesmo que em diferentes calibres (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

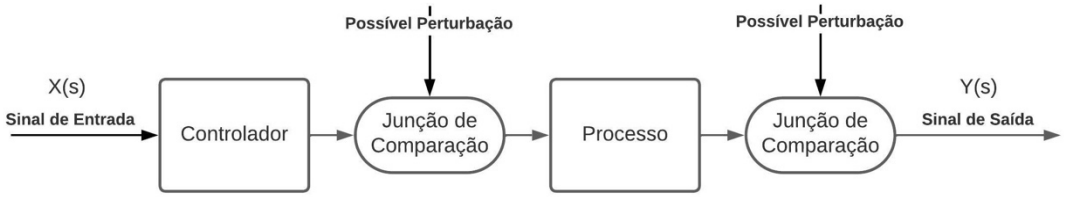
Os sistemas de controle geralmente assumem duas formas possíveis. A primeira forma é chamada de sistema de controle de malha aberta (Figura 32), enquanto a segunda forma é chamada de sistema de controle de malha fechada (Figura 33) (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

2.2.1 Sistemas de controle em malha aberta:

Essa modalidade de sistema de controle possui um sinal de entrada determinado com base na capacidade de tentativa e erro, ou seja, ele é estabelecido de acordo com a experiência do indivíduo responsável por defini-lo, uma vez que ele necessita de alteração em seu valor até o sinal de saída atingir o resultado desejado (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

Desse modo, fica implícita a sua incapacidade de compensar os efeitos de possíveis perturbações incidentes sobre o sinal de entrada pelo sistema, fazendo com que o sinal de saída apresente flutuações indesejadas (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

Figura 32 – Sistema de controle em malha aberta.



Fonte: Adaptado de Nise (2013-p36).

Nessa situação, um exemplo simples de sistema de controle em malha aberta seria um veículo, como um carro, uma moto, um caminhão, um ônibus, entre outros. O veículo, estando ligado, somente acelera, caso o pedal seja pressionado (MATHWORKS, 2023).

A pressão aplicada sobre o pedal, depende da força executada pelos pés do motorista, e consequentemente da sua necessidade. Se ele deseja ir mais rápido, aplica mais força e se deseja ir mais devagar, aplica menos força (BOLTON, 1995; MATHWORKS, 2023).

Logo, essa situação assume as características de um sistema de malha aberta. Como uma referência na estrada não é levada em consideração, por exemplo, o sistema não utiliza uma placa de sinalização, para que o motorista saiba o limite de velocidade com que o carro pode se locomover, o controlador se materializa na relação existente entre o cérebro e a força que se aplica no pedal (BOLTON, 1995; MATHWORKS, 2023).

Logo, não existe um dado, ou uma informação utilizada para verificar a diferença entre a velocidade do carro e a velocidade permitida na rodovia (BOLTON, 1995; MATHWORKS, 2023).

2.2.2 Sistemas de controle em malha fechada:

Essa modalidade de sistema de controle possui um sinal de entrada utilizado como sinal de referência, um sinal realimentado baseado no sinal de saída, e um comparador responsável por estabelecer um sinal de erro (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

Esse sinal de erro se materializa na diferença existente entre o sinal de entrada e o sinal realimentado. Desse modo, o objetivo é fazer com que ele tenda a um valor cada vez mais próximo de 0, pois isso significa que o sinal de saída está se aproximando do sinal de entrada, e consequentemente do valor desejado pelo indivíduo (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

Ou seja, ele é estabelecido de acordo com uma comparação responsável por defini-lo, uma vez que ele necessita de alteração em seu valor até o sinal de saída atingir o resultado desejado (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

Desse modo, fica implícita a sua capacidade de compensar os efeitos de possíveis perturbações incidentes sobre o sinal de entrada pelo sistema, fazendo com que o sinal de saída contorne flutuações indesejadas (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

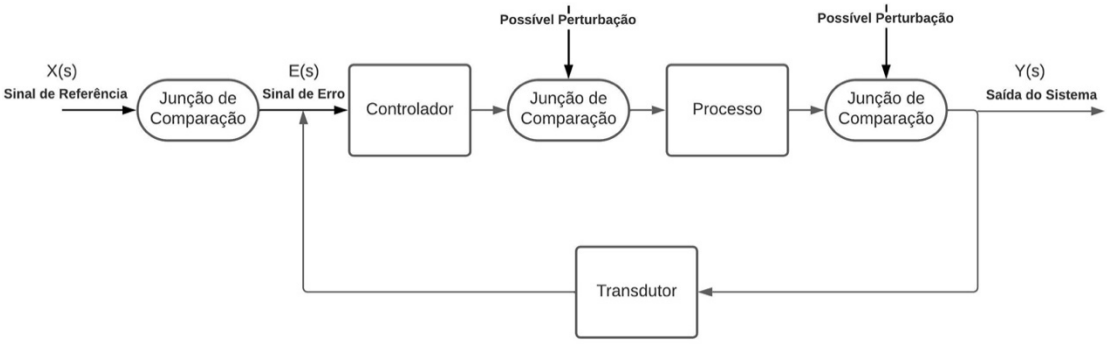
Ainda pensando no exemplo do carro, caso seja levado em conta a existência do velocímetro no painel do veículo, é possível reinterpretar esse sistema como um controlador de malha fechada (BOLTON, 1995; MATHWORKS, 2023).

Nessa situação, a sinalização visual é utilizada pelo motorista como uma forma de comparação entre o sinal de referência (placa de sinalização) e a velocidade real desenvolvida pelo carro (BOLTON, 1995; MATHWORKS, 2023).

Por meio do velocímetro é possível verificar se a saída do sistema está, ou não de acordo com o sinal de referência, e a possibilidade de existir uma diferença entre elas é o que origina o sinal de erro (BOLTON, 1995; MATHWORKS, 2023).

Consequentemente, o cérebro utiliza esse valor para determinar se é necessário pressionar mais, ou menos o pedal para chegar próximo do limite de velocidade da via (BOLTON, 1995; MATHWORKS, 2023).

Figura 33 – Sistema de controle de malha fechada.



Fonte: Adaptado de Nise (2013-p36).

2.2.3 Tipos de controlador:

Na literatura existem diversos tipos de controladores, que variam de acordo com a sua estrutura e abordagem de projeto. Contudo, dentre eles se destacam alguns tipos principais, sendo eles (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014):

- Controlador Proporcional (P): o controlador proporcional conduz o processo de controle por meio de um princípio de análise do erro. Geralmente, ele é caracterizado por uma relação proporcional à diferença entre o valor medido e o valor desejado, chamada de erro característico. Sendo assim, um sinal de controle é produzido com base nessa diferença, tornando-se atuante sobre a planta, ou processo. Essa modalidade de controlador é bastante simples, e sua principal desvantagem é a suscetibilidade a erros durante o regime transitório (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

- Controlador Proporcional-integral (PI): o controlador proporcional-integral também conduz o processo de controle por meio de um princípio de análise do erro. Contudo, além da relação proporcional à diferença entre o valor medido e o valor desejado, além de uma porção integral, atuando sobre essa mesma diferença, com base na passagem de tempo. O erro da leitura atual é avaliado com base no erro presente em amostras anteriores. Essa modalidade de controlador é mais precisa, e sua principal desvantagem é a suscetibilidade ao overshoot, ou seja, a somatização dos erros pode sofrer um crescimento exponencial, fazendo com que ela assuma uma tendência ao infinito. (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).
- Controlador Proporcional-Derivativo (PD): o controlador proporcional-derivativo apresenta um comportamento análogo ao controlador proporcional-integral. Contudo, nessa situação, além da relação proporcional à diferença entre o valor medido e o valor desejado, existe uma ação derivativa sobre essa mesma diferença, com base na passagem de tempo. O erro das leituras futuras é avaliado por meio do erro presente nas amostras atuais. Essa modalidade de controlador é precisa, contudo, seu principal objetivo é evitar a ocorrência de oscilações. Sua maior desvantagem é a suscetibilidade a ruídos (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).
- Controlador Proporcional-Integral-Derivativo (PID): o controlador proporcional-integral-derivativo é uma combinação do comportamento do controlador proporcional, integral e derivativo. É uma modalidade de controlador presente na indústria, pois as três características asseguram a capacidade de avaliar, o erro presente nas amostras atuais, passadas e futuras. Como resultado, ele é robusto e versátil, podendo atender a diversos requisitos de desempenho do sistema (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

- Controlador de lógica fuzzy: o controlador de lógica fuzzy utiliza uma abordagem baseada em um determinado conjunto de regras. Essas regras são composições linguísticas e servem de estrutura para a construção do sinal de controle desejado. Ele está voltado para aplicações onde os sistemas dinâmicos apresentam um comportamento não linear, isto é, tem um grau de imprecisão inerente a sua natureza. Porém, mesmo sendo adequados a esse tipo de sistema, sua calibração é difícil (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).
- Controlador adaptativo: o controlador adaptativo utiliza uma abordagem baseada em aprendizado de máquina, ou seja, Machine Learning. Consequentemente, não existe necessidade de ajuste manual dos parâmetros de calibração do controlador, pois ele possui capacidade para ajustar automaticamente os parâmetros, conforme mudanças significativas ocorrem no sistema (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

Como já destacado, o controlador a ser empregado nesse trabalho será o Controlador Proporcional-Integral-Derivativo (PID), como também suas respectivas variações. Mas, é importante salientar a importância de outras modalidades de controladores como o Controlador de lógica fuzzy e o Controlador adaptativo (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

Ou seja, cada tipo de controlador tem vantagens e desvantagens, e a escolha depende da aplicação específica. A escolha do controlador PID foi realizada devido a maior familiaridade com a sua respectiva composição, como também seu funcionamento, que será abordado em uma seção posterior deste trabalho, onde serão destacadas as especificações da aplicação, das características do sistema e dos requisitos de desempenho (BENNET, 1993; BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013; GOLNARIGHI; KUO, 2009; NISE, 2013; OGATA, 2014).

Cada termo do controlador PID possui uma influência específica sobre a saída do sistema em análise. Conforme, destacado anteriormente, o termo Proporcional (P) é a porção responsável por avaliar o erro presente na leitura atual realizada pelo sensor. Ele não sofre influências diretas da passagem do tempo, mas possui um ganho característico associado ao seu comportamento. A equação abaixo ilustra a influência dele sobre a saída (BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

$$y(t) = K_p \cdot e \tag{1}$$

Onde (t) é a variável representativa do tempo (em segundos), (y) é variável representativa da saída do sistema, (K_p) é a variável representativa do ganho proporcional e (e) representa o erro.

O termo Integral (I) é a porção responsável por avaliar o erro presente nas leituras passadas, buscando prever o erro presente na leitura atual. Enquanto, o termo Derivativo (D) é a porção responsável por avaliar o erro presente nas leituras futuras, se valendo do erro da amostra atual e das amostras passadas. Diferentemente do termo proporcional anterior, eles sofrem influências diretas da passagem do tempo e possuem ganhos característico aos seus comportamentos. As equações abaixo ilustram a influência deles sobre a saída (BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

$$y(t) = K_i \cdot \int_0^t e(t) \cdot dt \tag{2}$$

$$y(t) = K_d \cdot \frac{de(t)}{dt} \tag{3}$$

Onde (t) é variável tempo (y) é variável representativa da saída do controlador, (K_i) é variável representativa do ganho integral, (K_d) é variável representativa do ganho derivativo e $(e(t))$ é o erro em função do tempo.

Quando, a saída do controlador possui as influências de cada um dos termos, ou seja, das equações (1), (2) e (3), torna-se viável estabelecer uma equação característica, que assume uma composição generalizada, contabilizando cada um dos comportamentos (BOLTON, 1995; FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

$$y(t) = (K_p \cdot e(t)) + (K_i \cdot \int_0^t e(t)\, dt) + \ (K_d \cdot \frac{de(t)}{dt}) \tag{4}$$

Aplicando a Transformada de Laplace sobre a equação (4), obtém-se como resultado:

$$G_c(s) = \frac{Y(s)}{E(s)} = K_p + \frac{K_i}{s} + (K_d \cdot s) \tag{5}$$

Onde (s) é a variável de Laplace, $G_c(s)$ é a função de transferência do controlador, $\mathrm{E}(s)$ é a entrada do controlador, $\mathrm{Y}(s)$ é a saída do controlador, (K_p) é a variável representativa do ganho proporcional, (K_i) é variável representativa do ganho integral e (K_d) é a variável representativa do ganho derivativo.

2.3 DEFINIÇÃO DE ROBÔS:

Como destacado ao longo da introdução, o conceito de robô surgiu há muitos anos, e conforme o passar do tempo acabou sofrendo uma série de modificações. Porém, a fundamentação da principal visão da sociedade sobre robô foi bastante influenciada pela ficção científica. Também existem algumas definições mais recentes formalizadas por dicionários, e até mesmo livros.

Define-se "robô" como "um aparelho automático, com aspecto de boneco, capaz de executar tarefas desempenhadas por seres humanos", ou "Aparelho automático, com aspecto humanoide, capaz de se movimentar e executar diferentes tarefas", e até mesmo, como "Mecanismo cujo comando é controlado automaticamente" (DICIO, 2023; MICHAELLIS, 2023).

Já os autores preferem definir o conceito de robô como uma máquina com autonomia e programação, capaz de executar tarefas de forma automatizada, sem a necessidade de intervenção humana direta (CRAIG, 2005; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Os robôs são projetados para desempenhar uma ampla variedade de tarefas em diferentes ambientes e situações. Desde ambientes altamente controlados, como fábricas, até ambientes não controlados, como espaços subaquáticos ou planetas (CRAIG, 2005; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Em termos de composição, os robôs geralmente consistem de três componentes principais (CRAIG, 2005; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004):

- Um sistema de controle: que permite ao robô processar informações e tomar decisões (CRAIG, 2005; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004);
- Um conjunto de sensores: que fornece ao robô informações sobre o ambiente ao seu redor (CRAIG, 2005; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004);
- Um conjunto de atuadores: que permite ao robô realizar tarefas físicas, como mover objetos ou manipular equipamentos (CRAIG, 2005; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Além disso, muitos robôs estão equipados com recursos adicionais, como câmeras, sensores químicos e ferramentas para realizar exploração e coleta de dados (CRAIG, 2005; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Os robôs são projetados para desempenhar uma ampla variedade de tarefas, como soldagem, montagem, inspeção, transporte, limpeza entre outras. Sua adaptabilidade dele é uma consequência direta de sua capacidade de programação (CRAIG, 2005; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

A adaptabilidade proporciona uma notável gama de aplicações, seja para realizar movimentos repetitivos que podem ser monótonos para os seres humanos, ou para operar em ambientes perigosos que podem ser hostis para os trabalhadores. Como resultado, os robôs contribuem significativamente para melhorar a eficiência e a segurança no ambiente de trabalho (CRAIG, 2005; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

2.3.1 Classificação de robôs:

Atualmente, existe uma alta variedade de robôs presente no mercado. Dessa forma, eles acabam apresentando semelhanças e diferenças, permitindo classificá-los, como também separá-los em diferentes grupos (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Como mencionado anteriormente, um modo bastante recorrente é observar os robôs levando em consideração a função que eles são capazes de executar. Portanto, é possível identificar as seguintes tipologias (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004):

- Robôs industriais: são a categoria de robôs mais comuns no chão de fábrica. Eles são projetados com o objetivo de implementação nas linhas de produção, onde diversas tarefas ficam sobre sua responsabilidade. Nessa situação, as tarefas geralmente incluem a montagem de peças, pintura, soldagem, embalagem e manuseio de materiais. (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).
- Robôs móveis: são a categoria de robôs com capacidade de se movimentarem de modo autônomo. Ao serem controlados remotamente podem conduzir atividades nos mais diversos ambientes, enfrentando uma elevada quantidade de situações. Desse modo, sua aplicação está relacionada com a exploração em ambientes desconhecidos, operações de busca e resgate, entregas e inspeções (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).
- Robôs de serviço: é uma classe de robôs dedicados a realizarem atividades que necessitam de algum nível de interação com seres humanos. Desse modo, são utilizados para limpeza de pisos, assistência em hospitais e hotéis, como também cuidados para o público idoso (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

- Robôs militares: é uma classe de robôs dedicados a realizarem atividades, e concederem apoio as forças militares. São amplamente aplicados na detecção de explosivos, vigilância e reconhecimento (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).
- Robôs aéreos: simplificadamente são robôs que apresentam a capacidade de voar. O mais dinamizado no mercado atual são os drones. Os drones têm uma ampla variedade de aplicações, envolvendo áreas de inspeção, entregas, vigilância e fotografia aérea (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).
- Robôs submarinos: simplificadamente são robôs que apresentam a capacidade de se deslocarem debaixo d'água. Os principais exemplos envolvem os veículos de exploração e inspeção (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).
- Robôs colaborativos: são robôs voltados para trabalhar em e colaboração com seres humanos. Desse modo, devem seguir uma série de medidas de segurança, para que os trabalhadores consigam realizar suas tarefas. Sua aplicação está associada a processos produtivos que exigem habilidades complementares, como manipulação de objetos pesados ou precisão em cirurgias (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).
- Robôs educacionais: são robôs voltados para aplicação no ambiente educacional, envolvendo o ensino básico e o ensino superior. Se destacam por apresentar formas intuitivas de prototipagem e programação, favorecendo o ensino de robótica para crianças, jovens e adultos (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Existem também outras formas de classificação, como, por exemplo, de acordo com o tipo de movimento, a capacidade de interação, a estrutura física, e até mesmo o seu nível de complexidade. Dessa forma, outras classificações surgem (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004):

- Baseado no tipo de movimento: Nesse caso, a classificação leva como principal critério a forma de locomoção do robô e o ambiente ao qual ela está destinada. Os robôs são classificados em robôs terrestres, robôs aéreos, robôs submarinos, robôs com rodas, robôs com pernas, entre outros (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).
- Baseado na capacidade de interação: Nesse caso, a classificação leva como principal critério a maneira como robô consegue estabelecer uma interação com o ambiente ao seu redor. Em alguns casos, essa interação pode envolver o próprio ser humano. Portanto, as principais categorias são os robôs autônomos (operam sem intervenção humana), robôs semiautônomos (operam com supervisão humana), e robôs colaborativos (trabalham em conjunto com seres humanos) (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).
- Baseado na estrutura física: A estrutura física dos robôs é um critério bastante importante para segmentá-los em diferentes grupos. Como os robôs conseguem assumir uma grande variedade de formas e de tamanhos, classifica-los com base nas suas características físicas, isto é, as principais semelhanças existentes entre eles e os organismos a partir dos quais foram inspirados resulta no aparecimento de robôs humanoides (semelhantes a seres humanos), ou robôs de braço, também chamados de manipuladores mecânicos (semelhantes a partes do corpo humano) (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

- Baseado na complexidade: A complexidade dos robôs é outro critério bastante importante para segmentá-los em diferentes grupos. Como os robôs apresentam especificidades, eles conseguem se dedicar a execução de tarefas mais simples, ou de tarefas mais complexas. Desse modo, robôs simples se dedicam a realização de tarefas repetitivas, enquanto robôs mais complexos, se dedicam a realização de tarefas que exigem certo nível de autonomia, sendo os casos mais extremos associados ao uso de inteligência artificial e de aprendizado de máquina (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

O robô construído para o desenvolvimento desse trabalho, pensando na sua funcionalidade, poderia ser classificado como móvel, uma vez que ele apresenta a capacidade de se mover autonomamente para realizar tarefas em ambientes dinâmicos (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Entretanto, nada impossibilita estender essa classificação o fazendo se enquadrar como um robô terrestre (tipo de movimento), autônomo (capacidade de interação/complexidade), e até mesmo robô diferencial (estrutura física) (CRAIG, 2005; KERAMAS, 1998; LOZANO-PÉREZ, 1990; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

2.3.2 Especificação da classificação de robôs:

No caso dos robôs seguidores de linha, existe a preferência de alguns autores pela expressão “AGV” (Automated Guided Vehicle), empregada em português para designar um Veículo Guiado Automaticamente (NEHMZOW, 2002; SIEGWART; NOURBAKHSH, 2004).

Esse veículo é um tipo de robô móvel autônomo projetado para transportar cargas de forma automatizada em ambientes industriais, logísticos, ou até mesmo centros de distribuição, sendo encarregados pelo transporte de materiais, assim como peças (NEHMZOW, 2002; SIEGWART; NOURBAKHSH, 2004).

Os veículos guiados automaticamente são equipados com sensores, sistemas de navegação e controle, permitindo que eles se movam de maneira autônoma e evitem obstáculos, seguindo um conjunto predefinido de rotas, ou instruções de uma determinada missão (NEHMZOW, 2002; SIEGWART; NOURBAKHSH, 2004).

Conforme destacado anteriormente, eles são projetados para transportar cargas de diferentes tamanhos e pesos, exigindo o uso de baterias recarregáveis para sua alimentação (NEHMZOW, 2002; SIEGWART; NOURBAKHSH, 2004).

As baterias recarregáveis são uma ótima solução para viabilizar uma operação contínua e eficiente. Principalmente, pelo fato de alguns modelos de AGVs estabelecerem comunicação com outros robôs e sistemas de automação, assegurando uma produção mais coordenada e eficiente (NEHMZOW, 2002; SIEGWART; NOURBAKHSH, 2004).

Paralelamente, outros autores optam pela nomenclatura de robô diferencial. Essa nomenclatura é empregada para robôs móveis que usam duas rodas independentes para movimentação, em vez de um sistema de rodas omni ou pernas. Eles recebem esse nome, pois executam manobras de movimento diferencial, onde a velocidade de cada roda pode ser controlada independentemente, permitindo que o robô gire em torno de seu próprio eixo, realize curvas apertadas e navegue em terrenos irregulares. Além disso, o robô diferencial é capaz de avançar em linha reta, fazendo com que ambas as rodas girem na mesma velocidade. Este tipo de robô é comumente implementado em aplicações industriais, bem como em aplicações de robótica móvel, como robôs de limpeza, robôs de vigilância, robôs de resgate, entre outros (LOZANO-PÉREZ, 1990).

Mesmo possuindo algumas diferenças na classificação dessa categoria de robôs, devido a linhas de estudo distintas, principalmente no referencial utilizado como base, os autores apresentam unanimidade em destacarem a característica de não holonomia que será melhor definida na seção seguinte (LOZANO-PÉREZ, 1990; SIEGWART; NOURBAKHSH, 2004).

2.4 NÃO HOLONOMIA:

Holonomia é um conceito matemático, fundamentado muitas vezes na área da física, que está intimamente relacionado com à propriedade de uma curva, ou mais precisamente, à propriedade de uma trajetória de não possuir uma dependência ao caminho percorrido, mas sim as suas respectivas posições inicial e final no espaço (LOZANO-PÉREZ, 1990; SIEGWART; NOURBAKHSH, 2004).

Sendo assim, a holonomia é a propriedade responsável por estabelecer, que ao longo de uma dada trajetória fechada em um dado espaço, o objeto (móvel) responsável por percorrê-la, volta ao seu estado original independentemente do caminho percorrido (LOZANO-PÉREZ, 1990; SIEGWART; NOURBAKHSH, 2004).

O termo "não holonomia" é usado para descrever situações em que a propriedade da holonomia não é satisfeita, ou não é observada. Isso tem a possibilidade de acontecer, quando a trajetória depende do caminho percorrido. Isto é, quando o objeto se desloca ao longo da trajetória não retorna ao seu estado original após percorrê-la. Na física, um exemplo de sistema não holonômico é um pêndulo simples. O movimento de um pêndulo depende da trajetória percorrida, e a holonomia não é satisfeita. Outros exemplos de sistemas não holonômicos incluem sistemas com atrito, sistemas com restrições cinemáticas e sistemas com restrições geométricas complexas (LOZANO-PÉREZ, 1990; SIEGWART; NOURBAKHSH, 2004).

No caso dos robôs diferenciais, ou veículo guiado automaticamente, a não holonomia está associada a incapacidade de movimento em todas as direções possíveis, ou seja, apresentarem uma restrição relacionada ao seu grau de liberdade no espaço (LOZANO-PÉREZ, 1990; SIEGWART; NOURBAKHSH, 2004).

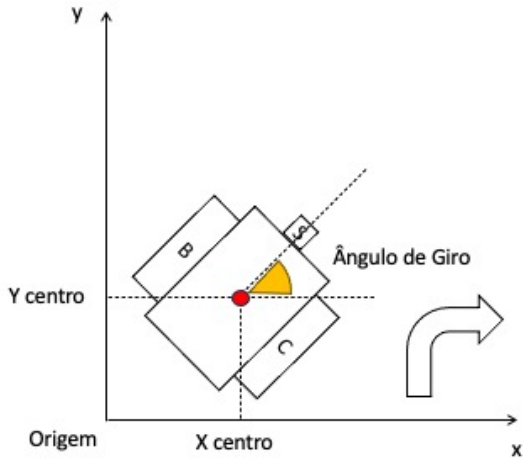
Desse modo, robôs holonômicos são aqueles que não apresentam restrições cinemáticas, enquanto robôs não holonômicos são aqueles que apresentam restrições cinemáticas (LOZANO-PÉREZ, 1990; SIEGWART; NOURBAKHSH, 2004).

Ao considerar o fundamento matemático do conceito, o funcionamento do robô ao se basear no princípio da diferenciação (ou derivação), não pode ser submetido ao princípio da integração para obter a sua posição espacial (LOZANO-PÉREZ, 1990; SIEGWART; NOURBAKHSH, 2004).

A Figura 34 ilustra esse conceito aplicado à curva executada por um robô. Nessa situação o simples ato de virar à direita, ou de virar à esquerda é uma composição de uma série de valores de tensão aplicados a cada um dos motores.

Logo, o robô não consegue virar normalmente, sendo necessário desenvolver o movimento em linha reta de um modo que o movimento da roda direita e o movimento da roda esquerda consigam executar uma compensação em sua trajetória, para que a curva seja concluída.

Figura 34 – Análise da curva executada por um robô não-homonômico.



Fonte: Adaptado de LOZANO-PÉREZ (1990-p.33-Fig3).

2.5 CONTROLADORES LÓGICOS PROGRAMÁVEIS E MICROCONTROLADORES:

Tanto os CLPs (Controladores Lógicos Programáveis), quanto os microcontroladores são classificados como dois grupos de dispositivos eletrônicos programáveis usados para controlar processos automatizados. No entanto, eles têm algumas diferenças importantes em termos de design, aplicação e capacidades (BANDEIRA; CARNEIRO, 2021; BARBOSA, 2017; JÚNIOR, 2018; MIRANDA, 2015; RIBEIRO, 2001).

Os CLPs são uma classe de dispositivos voltados para desempenhar o controle de processos de automação em ambientes industriais, tais como fábricas, refinarias, instalações de tratamento de água, entre outros. Eles são projetados para trabalhar em uma alta variedade de ambientes com características extremas, ou não, de alta temperatura, vibração, entre outros fatores que possam impactar na operação. Eles são compostos por uma CPU, entradas digitais e analógicas, saídas digitais e analógicas, além de interfaces de comunicação, tais como RS232, RS485, Ethernet e outros protocolos. Os programas envolvem linguagens baseadas na construção lógica de diagramas, como Ladder e Function Block. Essas linguagens são otimizadas para a construção de sistemas de controle e automação, como também possuem estruturas capazes de suportar rotinas em tempo real, com a finalidade de controlar processos com alta precisão e alta segurança (BANDEIRA; CARNEIRO, 2021; BARBOSA, 2017; JÚNIOR, 2018; MIRANDA, 2015; RIBEIRO, 2001).

Paralelamente, os microcontroladores são dispositivos eletrônicos menores e mais compactos. Sua principal vantagem é a alta versatilidade, favorecendo a sua aplicação em uma alta gama de processos, desde projetos simples, como sistemas de automação residencial e robótica educacional, até aplicações mais complexas, como equipamentos médicos, e controle de tráfego (BANDEIRA; CARNEIRO, 2021; BARBOSA, 2017; JÚNIOR, 2018; MIRANDA, 2015; RIBEIRO, 2001).

O microcontrolador é um sistema integrado com uma CPU, memória de programa, memória de dados, portas de entrada e saída, conversores analógico-digital, entre outros componentes e dispositivos. Eles são programados em linguagens de baixo nível, como C ou Assembly. Os códigos envolvem a execução de várias tarefas, como controle de motores, leitura de sensores, exibição de informações em displays (BANDEIRA; CARNEIRO, 2021; BARBOSA, 2017; JÚNIOR, 2018; MIRANDA, 2015; RIBEIRO, 2001).

2.6 KITS DE ROBÓTICA:

Nesta seção será construída uma visão geral sobre o surgimento dos primeiros kits de robótica, como também serão apresentadas as características gerais de algumas linhas de produto da LEGO, com foco para o LEGO® MINDSTORMS® Ev3.

2.6.1 Contexto histórico:

Na década de 1980, um novo marco foi alcançado dentro da robótica, principalmente no setor educacional. Durante essa década, algumas empresas passaram a elaborar os primeiros kits de robótica com o objetivo de dinamizar o estudo da disciplina para a educação básica, como também para a educação superior (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Vale ressaltar, que ao longo desse período grande parte dos robôs eram extremamente caros, assim como muito difíceis de programar, limitando o acesso dessas tecnologias aos estudantes e aos entusiastas da área (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Mas, com a chegada dos microcontroladores, como destacado no item anterior, assim como dos computadores de uso pessoal, a robótica passou a assumir um caráter mais acessível, acentuado com a chegada dos kits de robótica (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Uma das principais contribuições feitas nessa área foi conduzido pela empresa norte-americana Logo Computer Systems Incorporation. A empresa desenvolveu um software capaz de desenhar diversas formas geométricas (LOGO FOUNDATION, 2015)

Com o Logo Turtle Kit, um robô no formato de uma tartaruga, uma caneta retrátil na ponta permitia aos alunos levar a simulação das telas do computador, para folhas de papel. A linguagem de programação desenvolvida pela empresa chamou a atenção de outras companhias como a Apple, a IBM e até mesmo a LEGO, que em conjunto com o MIT, culminou na criação da linha MINDSTORMS (LOGO FOUNDATION, 2015).

Posteriormente, outros kits de robótica foram surgindo, como, por exemplo o K'NEX Control Tech, lançado no ano de 1992 pela empresa americana K'NEX Industries. Esse kit apresentava um conjunto de peças de montar, responsáveis por viabilizar as primeiras noções de prototipagem rápida (K'NEX GROUP,2023).

Consequentemente, eles asseguraram a elaboração de códigos de maneira mais versátil, favorecendo a aplicação em situações mais simples, isto é, deslocando objetos em um determinado espaço, ou até mesmo acedendo luzes conforme a necessidade (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

A partir desse momento, os kits de robótica foram implementados de um modo cada vez mais crescente no mercado. Isso possibilitou que as empresas empenhassem maiores esforços para criar diferenças significativas entre os produtos comercializados por outras marcas, mas também versões anteriores do mesmo conjunto (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Dessa maneira, ocorreu uma expansão deles para diversas regiões do mundo, devido a um barateamento de alguns modelos. Atualmente, eles se materializam como uma das principais formas de ensino prático de robótica em escolas e universidades de todo o mundo, contribuindo para a educação da próxima geração de cientistas, de engenheiros e de inventores no geral (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

Os kits de robótica são usados em muitos contextos diferentes, incluindo escolas, universidades, laboratórios de pesquisa e por entusiastas de robótica em casa. Eles são uma ferramenta poderosa para ensinar habilidades de resolução de problemas, pensamento crítico, trabalho em equipe e comunicação (CRAIG, 2005; KERAMAS, 1998; NEHMZOW, 2002; NIKU, 2001; SIEGWART; NOURBAKHSH, 2004).

2.6.2 Estrutura geral:

Os kits de robótica geralmente são constituídos de um conjunto de peças, componentes, dispositivos e ferramentas, capazes de garantirem aos usuários construir seus próprios robôs. Muitos tipos diferentes de kits de robótica estão disponíveis, envolvendo os mais variados graus de complexidade, tamanho, assim como finalidade (LEGO® Education WeDo 2.0, 2018; EDUCATIONAL; LEGO; 2022; “LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

Alguns foram projetados especificamente para o público infantil, contando com peças maiores e mais fáceis de manusear, enquanto outros são voltados para o público adulto e estudantes avançados, contando com peças menores e mais diversificadas (LEGO® Education WeDo 2.0, 2018; EDUCATIONAL; LEGO; 2022; “LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

Logo, podem apresentar componentes como motores, sensores, placas de controle, fios e baterias. Já outros também vêm com software de programação para auxiliar os usuários a controlar seus robôs, além de personalizar o comportamento desenvolvido (LEGO® Education WeDo 2.0, 2018; EDUCATIONAL; LEGO; 2022; “LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

2.6.3 Linhas de produto da LEGO:

Os kits de robótica da LEGO são um ótimo exemplo de simplicidade e de complexidade, que esses materiais podem assumir. A empresa fornece várias opções para o público infantil, como, por exemplo o LEGO WeDo. Essa linha é considerada a mais básica, pois foi projetada especificamente para crianças em idade escolar. Ela funciona como o primeiro contato com o mundo da robótica e da programação, pois inclui um software educacional interativo que ensina conceitos de programação, matemática e ciências em geral (LEGO® Education WeDo 2.0, 2018).

Ainda no ramo educacional, A LEGO® Education também possui modelos mais avançados, como o LEGO® Education SPIKE Prime, projetado para alunos do ensino médio, oferecendo recursos avançados de robótica e programação. Ele tem peças de LEGO®, motores, sensores e um hub programável compatível com um software de programação avançado dedicado (EDUCATIONAL; LEGO; 2022).

Porém, a linha de produtos empregada nesse trabalho é o LEGO® MINDSTORMS®, o modelo mais popular da LEGO®. Ele foi projetado para ser usado por pessoas de todas as idades, possuindo recursos mais complexos para o uso de universitários e o uso de entusiastas da robótica. Ele também conta com peças de LEGO®, motores, sensores e uma unidade de controle programável. O software fornecido apresenta fácil utilização, e concede aos usuários uma possibilidade de programação dos robôs usando uma interface gráfica simples ("LEGO MINDSTORMS EV3 Hardware Developer Kit", 2023).

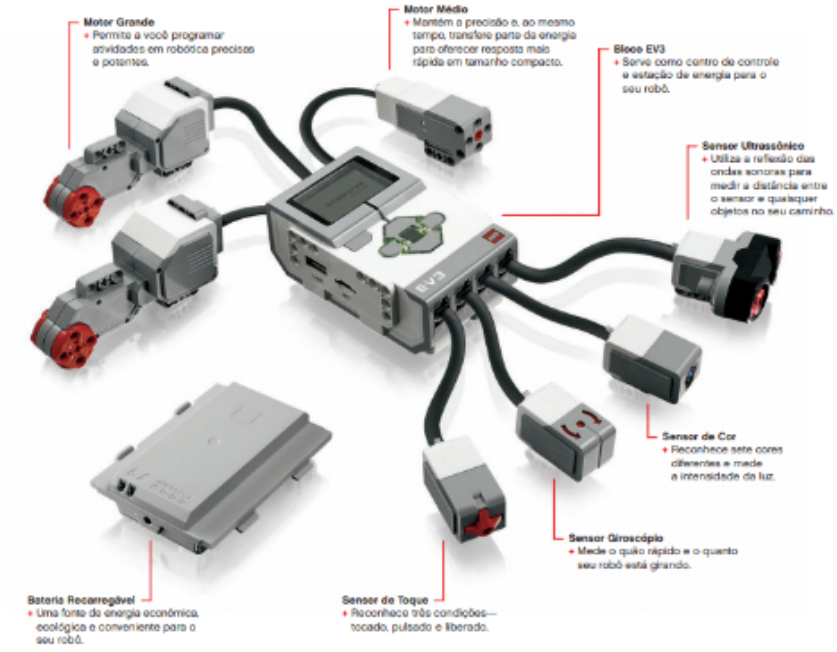
2.7 LEGO MINDSTORMS EV3:

Nessa seção o foco principal é destacar as características técnicas do kit de robótica LEGO® MINDSTORMS® Ev3. Desse modo, em primeiro lugar é apresentada a composição do Bloco Ev3, ou seja, da interface humano-máquina disponível. Posteriormente, uma breve descrição dos sensores e dos atuadores é feita, elencando cada uma de suas funcionalidades. Por último, existe uma breve descrição das peças disponíveis para prototipagem.

2.7.1 Apresentação:

O LEGO® MINDSTORMS® Ev3 (31313), lançado em janeiro de 2013, é a terceira geração de kits de robótica da LEGO® disponibilizados ao público geral. Ele possui como interface humano-máquina (IHM) o bloco Ev3 (em inglês Ev3 Brick), ou seja, um centro de controle que apresenta em sua constituição componentes eletrônicos de diversos tipos capazes de possibilitar uma alta variedade de funcionalidades conforme ilustrado pela Figura 35 ("LEGO MINDSTORMS EV3 Hardware Developer Kit", 2023).

Figura 35 – Visão geral do kit LEGO MINDSTORMS Ev3.



Fonte: Nitmaker (2023).

2.7.2 Características gerais:

Como processador principal, o bloco Ev3 apresenta um microcontrolador ARM9 de 32 bits, modelo AM1808 fornecido pela empresa Texas Instrument. Ao mesmo tempo, conta com uma memória DDR RAM de 64 MB, uma memória FLASH de 16 MB e uma memória EEPROM de 256 KB. Vale ressaltar que a memória pode ser estendida através do uso de um cartão Micro SDHC padrão de 32 GB (“LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

Em termos de comunicação com fio, é compatível com o USB 2.0 “Client Interface” por meio de uma porta de alta velocidade (480Mbit/s) e com o USB 1.1 “Host Interface” por meio de uma porta de velocidade máxima (12Mbit/s) (“LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

Essas comunicações permitem a conexão entre um computador de uso pessoal e o bloco Ev3, mas também a utilização de adaptadores USB para a conexão do bloco Ev3 em uma rede sem fio (“LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

Já, em termos de comunicação sem fio, é compatível com o Bluetooth, uma vez que conta nativamente módulo PAN1325 Bluetooth V2.1 EDR da Panasonic, constituído de um chip CC2550 da Texas Instrument, um conjunto Bluetooth modelo Blue Z, além do sistema de uso primário SPP (do inglês Serial Port Profile) (“LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

Sendo um dispositivo de entrada e saída (Input/Output), ele possui 4 portas de entrada compostas por uma conexão baseada na utilização de 6 fios. Os fios são compatíveis com interfaces de dados analógicos (0 – 5 Volts) e interfaces de dados digitais (“LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

Mais precisamente, as portas 1 e 2 apresentam comunicação UART de até 460 Kbits/s, enquanto as 3 e 4 apresentam comunicação UART de até 230 Kbits/s. As quatro portas de saída, também são compostas por uma conexão baseada na utilização de 6 fios, dedicados aos encoders e aos motores (“LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

Para facilitar a interação com o usuário existe um LCD integrado de 178x128 pixels, brancos e pretos, com a lógica de funcionamento baseada em uma matriz de ponto. Dessa forma, a área proporcionada tem dimensões de 29.9 x 41.1 mm. Também possui um alto-falante integrado, e seis botões do tipo Push-Button (“LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

Em termos de alimentação, o bloco Ev3 (Figura 36) tem um espaço reservado para 6 pilhas do tipo AA alcalinas, cada uma de 1.5 Volts totalizando uma exigência de 9 Volts. Sendo assim, elas podem ser substituídas por uma bateria de íon-lítio compatível com o compartimento disponível (“LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023)

Figura 36 – Bloco Ev3.

Fonte: Wskits - Educativos, Ciência, Robótica e Steam (2023).

2.7.3 Sensores:

Paralelamente, o conjunto conta com uma variedade de sensores, englobando as classes de sensores de cor, infravermelho e de toque.

O sensor de cor (Figura 37) possui três modalidades de funcionamento. O primeiro modo é chamado de sensor de luz, onde por meio da ativação do LED vermelho presente em sua estrutura, ele consegue medir a intensidade da luz refletida .

Já, o segundo modo é chamado de sensor ambiente, onde a ativação do mesmo LED vermelho é utilizada para medir a luz presente em seu entorno. Por último, o terceiro modo é chamado de sensor de cor, onde ele consegue detectar a coloração de um objeto colocado à sua frente (“LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

Figura 37 – Sensor de Cor.

Fonte: Wskits - Educativos, Ciência, Robótica e Steam (2023).

O sensor infravermelho (Figura 38) possui três modalidades de funcionamento também. Na primeira modalidade ele atua como um sensor de proximidade, na segunda como um sensor de rastreamento, e na terceira como um receptor de luz infravermelha (“LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

Figura 38 – Sensor Infravermelho.

Fonte: Wskits - Educativos, Ciência, Robótica e Steam (2023).

O sensor de toque (Figura 39) apresenta um funcionamento bastante simples, pois está condicionado aos dois estados mecânicos possíveis de serem assumidos, isto é, o estado pressionado e o estado não pressionado. Desse modo, geralmente é implementado com o objetivo de detectar algum tipo de obstáculo presente no ambiente (“LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

Figura 39 – Sensor de torque.

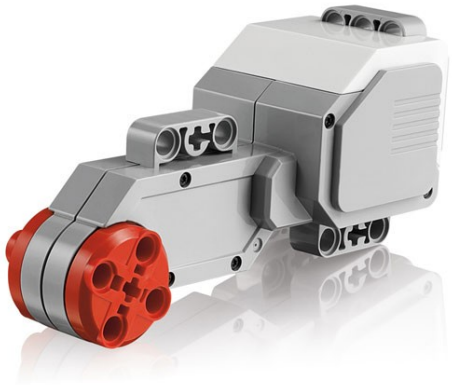
Fonte: Wskits - Educativos, Ciência, Robótica e Steam (2023).

2.7.4 Atuadores:

O kit incluí duas modalidades de motores, mais precisamente dois servos motores grandes (Figura 40) e um servo motor médio (Figura 41). Os servos motores grandes são mais parrudos e mais robustos, portanto, acabam sendo utilizados na movimentação ou no deslocamento dos robôs projetados (“LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

Por outro lado, os servos motores médios são mais acurados e mais precisos, portanto, acabam sendo utilizados na realização de ações mais rápidas ou mais velozes (“LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

Figura 40 – Servo motor grande.



Fonte: Wskits - Educativos, Ciência, Robótica e Steam (2023).

Figura 41 – Servo motor médio.

Fonte: Wskits - Educativos, Ciência, Robótica e Steam (2023).

2.7.5 Peças de prototipagem:

Em último lugar existe a disponibilização de uma alta variedade de peças, que asseguram a construção dos mais variados tipos de robôs, sejam eles pequenos, médios, ou grandes, com características próximas dos seres humanos (humanoides) ou de outros animais, e até mesmo instrumentos musicais (“LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

Além desses modelos, também existe a possibilidade de construir veículos, como por exemplo, aviões, barcos ou carros. (“LEGO MINDSTORMS EV3 Hardware Developer Kit”, 2023).

3. METODOLOGIA:

Com o objetivo de avaliar as principais contribuições desenvolvidas por outros estudantes e outros profissionais da área, foi conduzida uma pesquisa em algumas bases de dados para a seleção de um conjunto de documentos. Isto é, uma revisão sistemática baseada em artigos científicos apresentados em sites como Scopus, Science Direct, IEEE Xplorer, Scielo e Google Acadêmico.

Essa pesquisa resultou na definição de convenções de uma maneira mais coesa, além das etapas adotadas, para a realização dessa seção. Portanto, foi adotado um conjunto de palavras-chave com o intuito de limitar a quantidade de arquivos relacionados com o assunto tratado pelo trabalho em questão.

Sendo assim, a combinação utilizada foi “Lego” + “Mindstorms” + “Ev3” + “Robot” + “Line” + “Folllower”. A tabelas 1, 2, 3, 4 e 5 apresentam os resultados encontrados para cada um dos sites utilizados durante a busca, estabelecida em um intervalo de tempo dos últimos 5 anos, ou seja, de 2018 até 2022.

Paralelamente, definiu-se a possibilidade desses arquivos serem artigos de pesquisa e artigos de conferência, de preferência em inglês, em espanhol ou em português com acesso aberto, isto é, sem a necessidade de realizar um pagamento para a leitura do documento.

Tabela 1 – Resultados da pesquisa no site Scopus.

Combinação de Palavras	Quantidade de Resultados Encontrados
“Lego”	461
“Lego” + “Mindstorms”	41
“Lego” + “Mindstorms” + “Ev3”	28
“Lego” + “Mindstorms” + “Ev3” + “Robot”	20
“Lego” + “Mindstorms” + “Ev3” + “Robot” + “Line”	1
“Lego” + “Mindstorms” + “Ev3” + “Robot” + “Line” + “Follower”	0

Fonte: Autor.

Tabela 2 – Resultados da pesquisa no site Science Direct.

Combinação de Palavras	Quantidade de Resultados Encontrados
“Lego”	581
“Lego” + “Mindstorms”	38
“Lego” + “Mindstorms” + “Ev3”	8
“Lego” + “Mindstorms” + “Ev3” + “Robot”	8
“Lego” + “Mindstorms” + “Ev3” + “Robot” + “Line”	5
“Lego” + “Mindstorms” + “Ev3” + “Robot” + “Line” + “Follower”	4

Fonte: Autor.

Tabela 3 – Resultados da pesquisa no site IEEE Xplorer.

Combinação de Palavras	Quantidade de Resultados Encontrados
“Lego”	1030
“Lego” + “Mindstorms”	325
“Lego” + “Mindstorms” + “Ev3”	53
“Lego” + “Mindstorms” + “Ev3” + “Robot”	67
“Lego” + “Mindstorms” + “Ev3” + “Robot” + “Line”	1
“Lego” + “Mindstorms” + “Ev3” + “Robot” + “Line” + “Follower”	0

Fonte: Autor.

Tabela 4 – Resultados da pesquisa no site SciELO.

Combinação de Palavras	Quantidade de Resultados Encontrados
“Lego”	72
“Lego” + “Mindstorms”	4
“Lego” + “Mindstorms” + “Ev3”	2
“Lego” + “Mindstorms” + “Ev3” + “Robot”	1
“Lego” + “Mindstorms” + “Ev3” + “Robot” + “Line”	0
“Lego” + “Mindstorms” + “Ev3” + “Robot” + “Line” + “Follower”	0

Fonte: Autor.

Tabela 5 – Resultados da pesquisa no site Google Acadêmico.

Combinação de Palavras	Quantidade de Resultados Encontrados
“Lego”	3590
“Lego” + “Mindstorms”	358
“Lego” + “Mindstorms” + “Ev3”	111
“Lego” + “Mindstorms” + “Ev3” + “Robot”	105
“Lego” + “Mindstorms” + “Ev3” + “Robot” + “Line”	92
“Lego” + “Mindstorms” + “Ev3” + “Robot” + “Line” + “Follower”	46

Fonte: Autor.

Contudo, mesmo aplicando a combinação de palavras apresentada acima, houve algumas discrepâncias em relação aos documentos obtidos em cada um dos sites. Portanto, o resultado mais próximo do desejado ocorreu no site Science Direct, onde dos quatro arquivos, dois deles abrangiam todos os termos utilizados na combinação.

Mais precisamente, nos sites SciELO e IEEE Xplorer a busca não encontrou arquivos válidos, enquanto nos sites Scopus e Google Acadêmico, os arquivos obtidos estavam situados dentro da área de robótica envolvendo a aplicação do kit LEGO® MINDSTORMS® Ev3, contudo alguns dos estudos conduzidos não envolviam de fato a criação do robô seguidor de linha.

Buscando ampliar o número de documentos uma pesquisa paralela foi realizada com o objetivo de avaliar outros artigos científicos e alguns trabalhos de conclusão de curso relacionados ao assunto. Essa pesquisa paralela foi desenvolvida por meio de uma metodologia mais informal, sem a necessidade da condução de um procedimento robusto, como no caso do exemplo anterior.

Dessa forma, ao final do processo foram encontrados artigos tratando sobre robôs autônomos seguidores de linha, assim como planejamento de trajetória e planejamento de movimentação, artigos científicos conduzidos em nível de ensino técnico, artigos científicos conduzidos em nível de ensino superior, documentos de conferência ou simpósios de ensino de nível superior, além de trabalhos de conclusão de curso.

Desse modo, para organizar melhor os artigos selecionados os quadros seguintes foram criados:

Quadro 1 – Artigos científicos e documentos de conferência.

Nome do Artigo	Ano da Publicação	Nome dos Autores	Nome da Revista de Publicação	Principal Contribuição do Artigo Síntese
"On the Kinematics of Wheeled Mobile Robots"	1989	J. C. Alexander and J. H. Maddocks	International Journal of Robotic Research – IJRR	Apresenta uma perspectiva geral detalhada da cinemática de robôs móveis com rodas, útil na área de robótica móvel.
"Kinematic Modeling for Feedback Control of an Omnidirectional Wheeled Mobile Robot"	1987	P. Muir and C. Newman	Robotics and Automation, IEEE International Conference	Apresenta um modelo e um método de controle de feedback úteis para robôs móveis com rodas omnidirecionais, que podem ser empregados em diversas aplicações de robótica móvel.
"An Automatic Guidance System of a Self-Controlled Vehicle"	1897	T. Hongo and H. Arakawa G. Sugimoto and K, Tange and Y. Yamamoto	Industrial Electronics, IEEE Transactions	Apresenta uma abordagem prática para a implementação de um sistema de orientação automática para veículos autoguiados, podendo ser empregada em diferentes aplicações de robótica móvel.
"Local Path Control for an Autonomous Vehicle"	1988	W. Nelson and I. Cox	Robitics and Automation, IEEE International Conference	Apresenta uma abordagem prática para o controle de trajetória local em veículos autônomos, podendo ser empregada em diferentes aplicações de robótica móvel.
"Smooth Local Path Planning for Autonomous Vehicles"	1987	Y. Kanayama and B. Hartman	International Journal of Robotics Research	Apresenta uma abordagem prática para o planejamento de trajetória suave em veículos autônomos, podendo ser empregado em diferentes aplicações de robótica móvel.
"Low Pass Filter Applied to Color Sensor of Line Follower Robot"	2019	Ari Aharari, Yuka Ueda	Procedia Computer Science	Apresenta uma abordagem prática para melhorar a precisão do

				sensor de cor em robôs seguidores de linha, podendo ser empregado em diferentes aplicações de robótica móvel.
“Gathering of robots in a ring with mobile faults”	2018	Shantanu Das, Riccardo Focardi, Flaminia L. Luccio, Euripides Markou, Marco Squarcina	Theorical Computer Science	Apresenta uma abordagem para resolver o problema de aglomeração de robôs em um anel com falhas móveis, podendo ser empregado diferentes aplicações de robótica móvel.
“Seguidor de Linha Para LEGO ® Mindstorms Utilizando Controle PID”	2014	Robison C. Brito, Emanoeli Madalosso, Geovane A. O. Guibes	Computer on the Beach	Apresenta uma abordagem prática para o controle de movimento de robôs seguidores de linha utilizando controle PID, podendo ser empregado em diferentes plataformas de robótica móvel.
“Controlador PID Aplicado ao Rastreamento de Trajetória”	2019	Amanda Hikari Silva Tanaka, Ayllon Torres Scherwinski e Maximilian Jaderson de Melo	Mostra Nacional de Robótica – MNR	Apresenta uma aplicação prática do controlador PID em robótica, demonstrando sua efetividade no rastreamento de trajetórias pré-determinadas.
“Controlador PID Aplicado a Robótica Móvel”	2016	Thais Julia Borges Ribeiro	Mostra Nacional de Robótica – MNR	Apresenta uma revisão bibliográfica sobre o uso do controlador PID em robótica móvel, destacando suas vantagens e limitações na busca por soluções para os desafios apresentados.
“Projeto de Construção e Controle PID de um Manipulador Robótico para Auxílio na Aprendizagem		Felipe Castro Teixeira de Carvalho, Gabriela Lígia Reis,	Revista SBA: Controle e Automação	Apresenta o projeto, construção e controle de um manipulador robótico de baixo custo para uso em sala de

de Alunos de Graduação em Engenharia”	-	Luis Fernando Freire de Souza, Márcio Falcão Santos Barroso		aula, com o objetivo de auxiliar na aprendizagem de estudantes de engenharia na aplicação do controlador PID em robótica educacional, mostrando sua efetividade no controle de movimentos do robô.
“Controlador PID Aplicado a Programação de Robô Móvel”	2016	Thais Julia Borges Ribeiro, Masamori Kashiwagi	Congresso de Inovação, Ciência e Tecnologia do IFSP	Apresenta uma aplicação prática do controlador PID, demonstrando sua efetividade no controle de movimento de um robô móvel, buscando por soluções para desafios relacionados ao controle de movimento em robótica móvel.
“Controlador PID em Robô Seguidor de Linha em Plataforma LEGO EV3”	2019	Ana Beatriz Montenegro, Morgana Fernandes, Jonathas Wesley Bonfim, Robson Júnior e Vitor O. S. T. de Souza.	Simpósio de Engenharia, Gestão e Inovação – Sengi	Apresenta a implementação do controlador PID em um robô seguidor de linha utilizando a plataforma LEGO EV3, destacando sua importância no controle de posição e velocidade do robô, buscando por soluções para desafios relacionados ao seguimento de trajetórias em robótica.

Fonte: Autor.

Quadro 2 – Trabalhos de Conclusão de Curso.

Título do Trabalho de Conclusão de Curso	Ano da Apresentação	Nome dos Autores	Instituição de Ensino Superior	Principal Contribuição do Trabalho Síntese
"Desenvolvimento e Cooperação de Robôs Através da Plataforma Arduino"	2018	Volnei Fontana Júnior	Universidade Federal de Santa Catarina – Centro Tecnológico Departamento de Automação e Sistemas	Apresenta um estudo sobre a construção e programação de robôs utilizando a plataforma Arduino, destacando a importância da cooperação entre eles para a realização de tarefas complexas.
"Projeto Básico de Robô Seguidor de Linha Controlado por Arduino"	2021	Matheus Santos Bandeira e Rogério da Silva Carneiro	Universidade Federal Fluminense – Escola de Engenharia	Apresenta um estudo sobre a construção e programação de um robô seguidor de linha utilizando a plataforma Arduino, destacando a importância do uso de robôs deste tipo em diversas áreas.
"Robô LEGO NXT Seguidor de Trajetória com Controle Fuzzy"	2017	Alan Cardoso Barbosa	Universidade Estadual de Londrina Centro de Tecnologia e Urbanismo Departamento de Engenharia Elétrica	Apresenta um estudo sobre a construção e programação de um robô seguidor de trajetória utilizando a plataforma LEGO NXT e um controlador Fuzzy, destacando a importância do uso de técnicas de controle avançado em robótica.
"Comparação entre Controladores Clássicos e Fuzzy Comandando um Pêndulo Invertido Tipo Carro Implementado com LEGO MINDSTORM EV3"	2015	Arthur Reis Lara Miranda	Universidade Federal de Ouro Preto – Escola de Minas Gerais Colegiado do Curso de Engenharia de Controle e Automação - CECAU	Apresenta uma comparação entre os controladores clássicos e Fuzzy aplicados a um modelo de carro com pêndulo invertido construído com a plataforma LEGO Mindstorm EV3, mostrando que o controlador Fuzzy obteve melhores resultados em termos de desempenho e estabilidade.

Fonte: Autor.

3.1 ROBÔ SEGUIDOR DE LINHA:

Nessa seção é conduzida uma descrição sobre o robô seguidor de linha, destacando o seu conceito base, o processo de montagem do protótipo, as diferentes partes do protótipo, algumas especificidades presentes em sua estrutura e a uma adaptação feito pelo autor na montagem, para que fosse possível implementar o sensor de luz.

3.1.1 Conceito Base

A construção do protótipo foi feita utilizando os recursos de montagem disponíveis no site oficial da empresa LEGO®, dedicadas ao kit de robótica MINDSTORSMS ® Ev3 – 31313 (“Building instructions for 31313, LEGO® MINDSTORMS® EV3, LEGO® MINDSTORMS®”, 2013).

No site, existe uma seleção de 17 modelos básicos, englobando os mais diversos formatos e tamanhos, contudo como o objetivo é analisar o comportamento associado a um robô autônomo seguidor de linha, o modelo selecionado foi BOBB3E (“Building instructions for 31313, LEGO® MINDSTORMS® EV3, LEGO® MINDSTORMS®”, 2013).

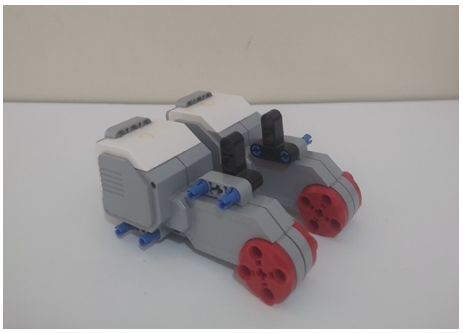
Esse robô conta com instruções de montagem e programação desenvolvidas por Kenneth R. Madsen, um engenheiro de tecnologia da informação associado ao grupo LEGO® (“Building instructions for 31313, LEGO® MINDSTORMS® EV3, LEGO® MINDSTORMS®”, 2013).

O conjunto de imagens abaixo ilustra todo o procedimento conduzido para a obtenção do protótipo em questão. A montagem teve em média uma duração de 2 horas e 30 minutos, mas esse tempo poderia ser facilmente reduzido, já que o processo foi realizado de modo lento, buscando sempre se atentar aos procedimentos de montagem exigidos em cada etapa.

Outro detalhe é que as peças solicitadas foram separadas conforme houvesse necessidade, e por apresentarem diferentes formas e tamanhos, acabaram tomando intervalos de tempo variados para serem encontradas.

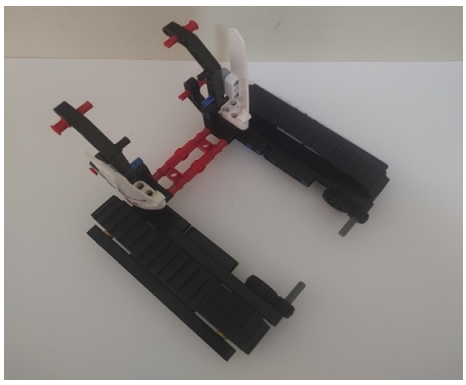
A montagem completa do protótipo está baseada na anexação de estruturas, ou seja, o conjunto dos dois servos motores grandes (Figura 42), o conjunto das rodas e do suporte do bloco Ev3 (Figura 43), o conjunto do servo motor médio e do sensor infravermelho (Figura 44), como também o bloco Ev3 (Figura 45).

Figura 42 – Conjunto dos dois servos motores.



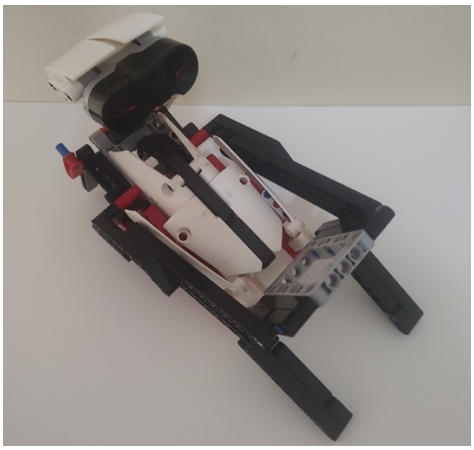
Fonte: Autor

Figura 43 – Conjunto rodas e suporte do bloco Ev3.



Fonte: Autor

Figura 44 – Conjunto do servo motor médio e do sensor infravermelho.



Fonte: Autor.

Figura 45 – Bloco Ev3.

Fonte: Autor.

3.1.2 Explicação do funcionamento do hardware

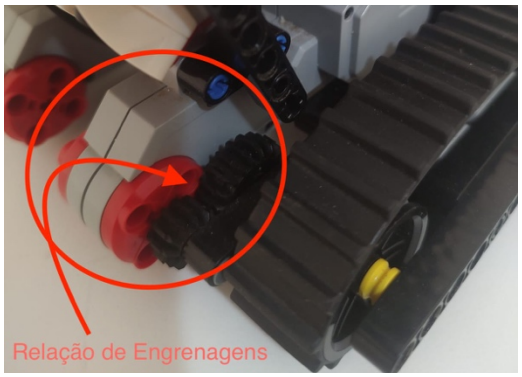
Conforme ilustrado pela Figura 42, o robô possui em sua estrutura dois motores trabalhando em conjunto, conectados as entradas B e C do Bloco Ev3. Cada um dos motores está conectado a extremidade interna de um eixo, enquanto a extremidade oposta está associada a uma roda.

Entre eles existe a presença de duas engrenagens estabelecendo uma relação, uma vez que elas apresentam uma quantidade de dentes diferentes.

Figura 46 – Relação de engrenagens no lado direito do robô.

Fonte: Autor.

Figura 47 – Relação de engrenagens no lado esquerdo do robô.



Fonte: Autor.

Nesse caso, a engrenagem conectada diretamente ao motor, ou seja, a engrenagem motora possui 12 dentes. Em contrapartida a outra engrenagem não conectada diretamente ao motor, ou seja, a engrenagem movida, possui 20 dentes.

Nessa situação, a ação resultante estabelece uma redução na velocidade da engrenagem motora, fazendo com que ela, em contrapartida, ganhe força.

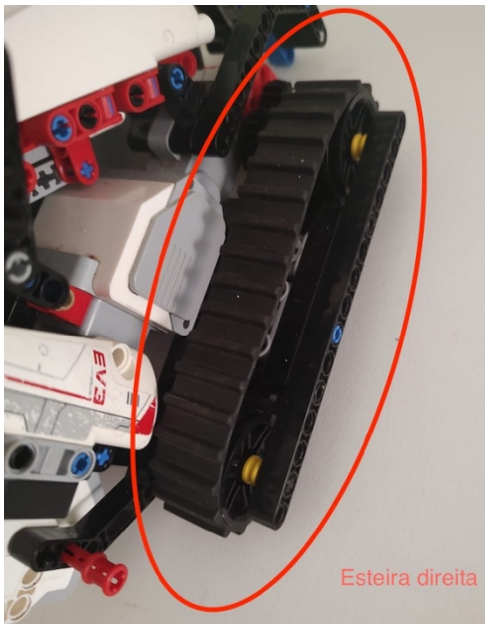
$$\frac{Dentes\ da\ Engrenagem\ Movida}{Dentes\ da\ Engrenagem\ Motora} = \frac{20}{12} = 1{,}6667 \tag{6}$$

Ao analisar a relação apresentada acima pela equação, fica mais evidente que a engrenagem motora precisa girar aproximadamente duas vezes, para que a engrenagem movida execute um giro. Portanto, destacando novamente a redução de velocidade.

As rodas pertencem a duas estruturas laterais anexas a porção central do robô, onde é possível identificar sua respectiva conexão a outra roda por meio de uma correia emborrachada, atuando na forma de pneu. Desse modo, esses robôs diferenciais, se movimentam com o emprego de esteiras.

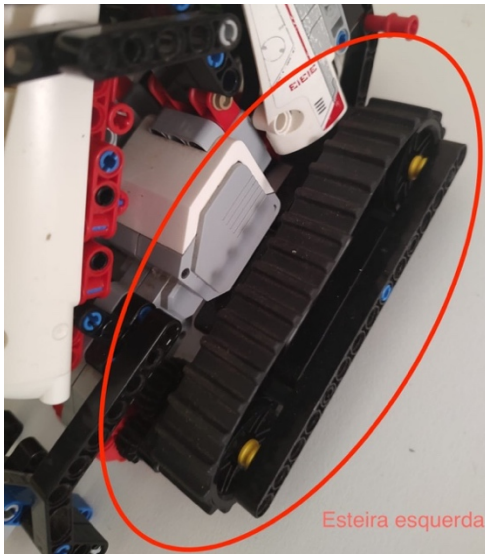
Logo, as estruturas laterais são interpretadas como um sistema de duas polias conectadas através de uma correia. Sendo assim, as rodas atuam como polias, enquanto a esteira emborrachada atua como correia. A Figura 48 e a Figura 49 ilustram a composição da esteira direita e da esteira esquerda, respectivamente:

Figura 48 – Esteira direita.



Fonte: Autor

Figura 49 – Esteira esquerda.



Fonte: Autor.

Nesse caso, as rodas assumem uma mesma velocidade linear, devido a conexão com a esteira de tamanho fixo, mas também uma mesma velocidade angular, pois elas possuem o mesmo raio. A Equação 7 e a Equação 8 apresentadas a seguir representam essa ideia:

$$\frac{n_1}{n_2} = \frac{D_2}{D_1} \rightarrow D_1 = D_2 \tag{7}$$

$$\frac{n_1}{n_2} = \frac{D_1}{D_1} = 1 \rightarrow \ \therefore n_1 = n_2 \tag{8}$$

Onde (n_1) é a variável responsável por representar o número de dentes da engrenagem associada a roda esquerda, (n_2) é a variável responsável por representar o número de dentes da engrenagem da roda direita, (D_1) representa o diâmetro da esteira presente na roda esquerda e (D_2) representa o diâmetro da esteira presente na roda direita.

Na seção central do robô, na porção superior existe o sensor infravermelho, como também uma plataforma elevadora e abaixadora, com seus movimentos coordenados pela presença do servo motor médio submetido a uma relação de engrenagens (Figura 50).

Figura 50 – Relação de engrenagens motor médio.

Fonte: Autor.

Essa relação envolve uma engrenagem motora com uma quantidade de dentes igual a 12, e uma engrenagem movida, com uma quantidade de dentes igual a 36. Desse modo, a relação de engrenagens também estabelece uma redução na velocidade da engrenagem motora, fazendo com que ela ganhe força.

$$\frac{Dentes\ da\ Engrenagem\ Movida}{Dentes\ da\ Engrenagem\ Motora} = \frac{36}{12} = 3 \tag{9}$$

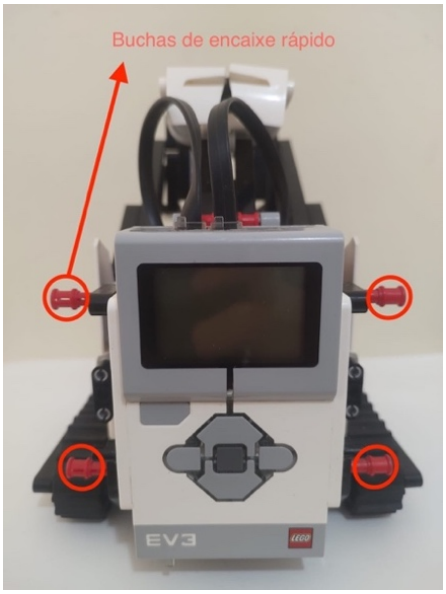
Ao analisar a relação apresentada acima na equação acima, fica mais evidente que a engrenagem motora precisa girar aproximadamente três vezes, para que a engrenagem movida execute um giro. Portanto, destacando novamente a redução de velocidade.

Na seção traseira o bloco Ev3 (Figura 51) está conectado de acordo com uma estrutura de encaixe rápido, isto é, as buchas vermelhas permitem retirá-lo com certa facilidade, caso seja necessário trocar a fonte de alimentação.

Já, na seção dianteira (Figura 52) foi posicionado um sensor de cor por meio de uma estrutura desenvolvida pelo próprio autor, isto é, a montagem original não contava com ele.

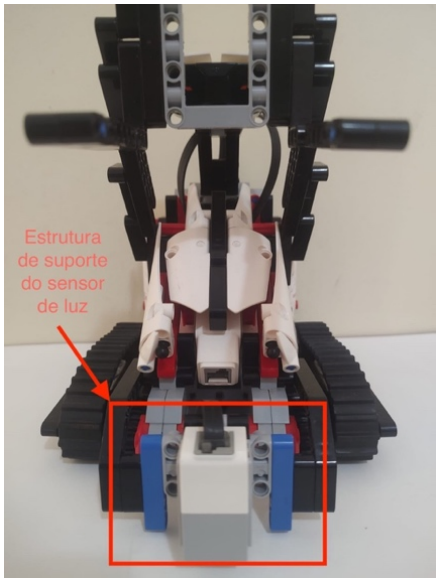
Porém, existia possibilidade de adicioná-lo sem grandes dificuldades, fator responsável por facilitar a criação do robô seguidor de linha, propriamente dito.

Figura 51 – Seção traseira do robô.



Fonte: Autor.

Figura 52 – Seção dianteira do robô.



Fonte: Autor.

3.1.3 Explicação do desenvolvimento do software:

O primeiro desafio enfrentado para a construção do controlador PID digitalmente foi a questão da compatibilidade entre o hardware utilizado com os softwares de desenvolvimento disponíveis.

O kit de robótica empregado para a montagem do protótipo é compatível com diversas versões da IDE fornecida pela LEGO®. Mas, tanto o ambiente voltado para o uso doméstico, quanto o ambiente voltado para o uso escolar apresentam algumas restrições, que impedem o desenvolvimento do controlador PID de uma forma simples e direta (“Firmware Developer Kit”, 2023; “Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython”, 2023).

Como os ambientes são baseados em uma linguagem de programação voltada para diagrama de blocos sem a possibilidade de alterar algumas propriedades em suas respectivas construções, a criação de lógicas relacionadas ao funcionamento dos motores e dos sensores não muito é versátil (“Firmware Developer Kit”, 2023; “Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython”, 2023).

Sendo assim, a criação, declaração e manipulação de variáveis é pouco intuitiva. Portanto, como alternativa fornecida pela LEGO® existe a possibilidade de tornar o hardware utilizado compatível com a linguagem de programação Python (“Firmware Developer Kit”, 2023; “Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython”, 2023).

Esse processo necessita da realização de um procedimento, relativamente trabalhoso, envolvendo a substituição do firmware incluído nativamente no bloco Ev3 por outra versão, chamada de firmware do desenvolvedor (“Firmware Developer Kit”, 2023; “Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython”, 2023).

A versão do desenvolvedor é disponibilizada no próprio site da LEGO®. A empresa é receptiva na questão de recursos “Open Source” (recursos de desenvolvimento aberto) disponibilizando uma ampla variedade de materiais (“Firmware Developer Kit”, 2023; “Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython”, 2023).

Dentre esses materiais, o essencial para a realização é o firmware do desenvolvedor fornecido pela equipe ev3dev, além da extensão Ev3 MicroPython e a biblioteca pybricks (“Firmware Developer Kit”, 2023; “Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython”, 2023).

3.1.4 Procedimento para instalação da versão do desenvolvedor

Como requisitos básicos do procedimento, um computador com sistema operacional Windows, ou com sistema operacional Machintosh é requirido, assim como uma conexão válida à internet, com acesso aos privilégios concedidos pelo administrador (“Firmware Developer Kit”, 2023; “Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython”, 2023).

Esses requisitos básicos são exigências para o procedimento de instalação. Com cada programa já instalado na máquina utilizada, não é necessário conduzir nenhum processo paralelo, ou seja, a criação e simulação dos códigos é feita facilmente pelo usuário (“Firmware Developer Kit”, 2023; “Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython”, 2023).

Entretanto, um cartão microSD com capacidade mínima de 4GB e capacidade máxima de 32GB, pertencente a Classe A1, é solicitado (o modelo microSDHC também é compatível). Paralelamente, um cabo mini-USB é necessário, para que seja possível baixar e simular os códigos no protótipo. Esses materiais estão presentes na Figura (53) (“Firmware Developer Kit”, 2023; “Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython”, 2023).

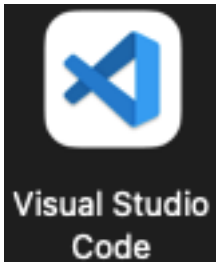
Figura 53 – Cabo micro USB e cartão microSD.

Fonte: Autor.

Também é recomendável baixar uma plataforma de desenvolvimento de códigos compatível com a extensão MicroPython. Desse modo, o Visual Studio Code (Figura 54) foi instalado, uma vez que ele é a opção sugerida pelo guia consultado (“Firmware Developer Kit”, 2023; “Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython”, 2023).

Posteriormente, a configuração básica e a configuração avançada foram conduzidas, sobrando apenas a preparação do cartão micro (“Firmware Developer Kit”, 2023; “Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython”, 2023).

Figura 54 – Ícone do software Visual Studio Code.



Fonte: Autor.

Conforme, destacado anteriormente, a extensão MicroPython compatível com Visual Studio Code exige que o bloco Ev3 esteja com seu firmware na versão do desenvolvedor (“Firmware Developer Kit”, 2023; “Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython”, 2023).

Dessa forma, um arquivo no formato “imagem de cartão” compactado, fornecida pela ev3dev (Figura 55), foi baixado e instalado no cartão microSD por meio do software Etcher (Figura 56) (“Firmware Developer Kit”, 2023; “Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython”, 2023).

Figura 55 – Ícone da extensão ev3dev presente no Visual Studio Code.

Fonte: Autor.

Figura 56 – Ícone do software Etcher.

Fonte: Autor.

O software Etcher converte esse arquivo de aproximadamente 360 MB no formato (.zip) em formato de “imagem de cartão” adequado, isto é, realiza uma ação conhecida como “flash” (“Firmware Developer Kit”, 2023; “Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython”, 2023).

3.1.5 Matlab e SIMULINK:

Inicialmente, toda a construção dos programas seria conduzida no Visual Studio Code, mas a biblioteca pybricks possui apenas capacidade de implementar os valores dos ganhos associados ao controlador PID. Logo, todo o processo de modelamento matemático dos motores e dos sensores acabaria substituído por uma simulação submetida ao método de tentativa e erro.

Buscando evitar essa situação, foi identificado que o software Matlab possui um conjunto de extensões, e módulos extras, capazes de adicionar ao recurso SIMULINK uma variedade de ferramentas compatíveis com o bloco Ev3, e capazes de assegurar o modelamento matemático desejado.

O Matlab não foi escolhido como primeira opção, pois o modelamento matemático exige o conhecimento de alguns parâmetros específicos de cada um dos componentes e dispositivos presentes no protótipo.

Porém, a LEGO® fornece um conjunto de informações muito limitado nesse aspecto, pois a versão de desenvolvedor dos documentos de hardware e software não contam com as características elétricas, como também com as características mecânicas dos motores e dos sensores.

Sendo assim, um sistema de aquisição de dados é necessário com o objetivo de estimar esses parâmetros, possibilitando a determinação de cada uma das equações matemáticas. Esse sistema de aquisição de dados é criado facilmente tanto no Visual Studio Code, quanto no Matlab, mas o grande desafio é fazer a simulação.

Enquanto o Visual Studio Code exige apenas o cabo micro-USB para a simulação, o Matlab exige um adaptador Wi-Fi (do inglês dongle), uma vez que o "external mode" não suporta a transmissão de informações pelo cabo micro-USB.

Nesse caso, o único adaptador Wi-Fi compatível com o bloco-Ev3 é o Netgear N150, um dispositivo de uma linha de produtos fora do mercado atualmente. Em sites de e-comerce, ou marketplace, ainda é possível encontrar alguns exemplares a venda, porém os custos são elevados. O produto se tornou bastante raro, como também desejado por diversos entusiastas de tecnologia.

Como opção, existem outros adaptadores compatíveis com o sistema operacional do bloco Ev3, como, por exemplo o adaptador USB com antena integrada (Figura 57), e o adaptador USB/rj45.

Esses dois modelos foram comprados e testados, mas não são suportados pelo firmware nativo do bloco Ev3. Já, com o firmware do desenvolvedor instalado no cartão micro-USB e plugado no bloco Ev3, eles funcionaram sem grandes problemas.

Figura 57 – Adaptador Wi-Fi adquirido.

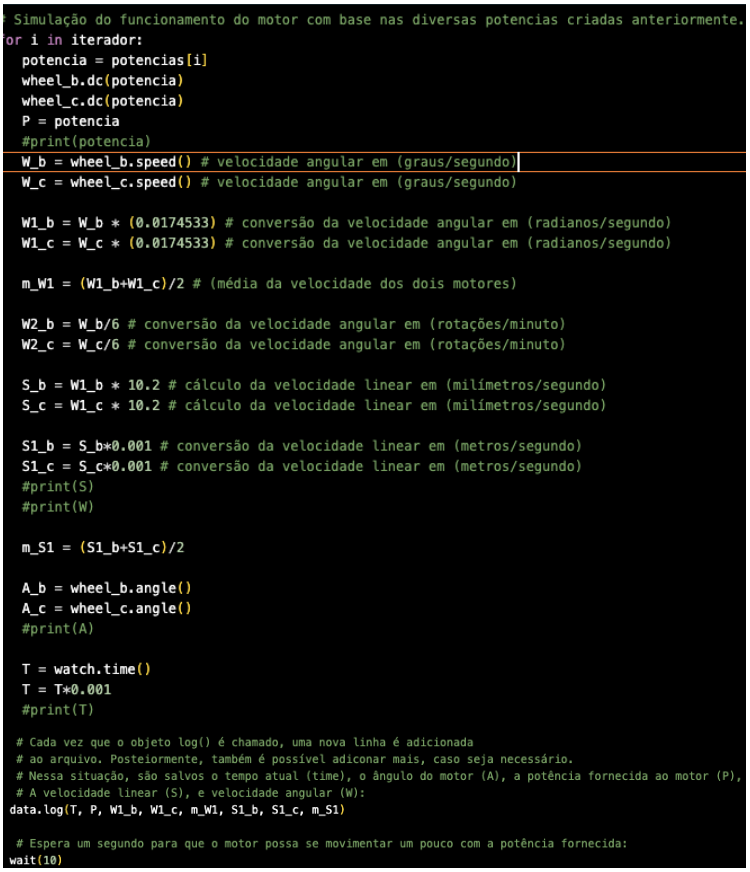
Fonte: Autor.

Uma série de testes foi conduzida, e se identificado que o Matlab não consegue se comunicar com o bloco Ev3 enquanto o firmware do desenvolvedor está em funcionamento, sua comunicação é viabilizada apenas com o firmware nativo.

Dessa maneira, a melhor alternativa foi criar o sistema de aquisição de dados no Visual Code Studio, e os modelos matemáticos no Matlab.

3.1.6 Criação do Sistema de Aquisição de Dados:

Figura 58 – Trecho do código responsável por criar o sistema de aquisição de dados.



```
# Simulação do funcionamento do motor com base nas diversas potencias criadas anteriormente.
for i in iterador:
  potencia = potencias[i]
  wheel_b.dc(potencia)
  wheel_c.dc(potencia)
  P = potencia
  #print(potencia)
  W_b = wheel_b.speed() # velocidade angular em (graus/segundo)
  W_c = wheel_c.speed() # velocidade angular em (graus/segundo)

  W1_b = W_b * (0.0174533) # conversão da velocidade angular em (radianos/segundo)
  W1_c = W_c * (0.0174533) # conversão da velocidade angular em (radianos/segundo)

  m_W1 = (W1_b+W1_c)/2 # (média da velocidade dos dois motores)

  W2_b = W_b/6 # conversão da velocidade angular em (rotações/minuto)
  W2_c = W_c/6 # conversão da velocidade angular em (rotações/minuto)

  S_b = W1_b * 10.2 # cálculo da velocidade linear em (milímetros/segundo)
  S_c = W1_c * 10.2 # cálculo da velocidade linear em (milímetros/segundo)

  S1_b = S_b*0.001 # conversão da velocidade linear em (metros/segundo)
  S1_c = S_c*0.001 # conversão da velocidade linear em (metros/segundo)
  #print(S)
  #print(W)

  m_S1 = (S1_b+S1_c)/2

  A_b = wheel_b.angle()
  A_c = wheel_c.angle()
  #print(A)

  T = watch.time()
  T = T*0.001
  #print(T)

 # Cada vez que o objeto log() é chamado, uma nova linha é adicionada
 # ao arquivo. Posteiormente, também é possível adiconar mais, caso seja necessário.
 # Nessa situação, são salvos o tempo atual (time), o ângulo do motor (A), a potência fornecida ao motor (P),
 # A velocidade linear (S), e velocidade angular (W):
data.log(T, P, W1_b, W1_c, m_W1, S1_b, S1_c, m_S1)

 # Espera um segundo para que o motor possa se movimentar um pouco com a potência fornecida:
wait(10)
```

Fonte: Autor

O sistema de aquisição de dados (Figura 58) foi criado no Visual Studio Code a partir do modelo de código conhecido por “Datalog”. Esse modelo permite obter os dados de um motor, e até mesmo de um sensor, em funcionamento e armazená-los em um documento no formato (.csv). O código completo e comentado está presente na seção Apêndice A.

Basicamente, os servos motores grandes do kit de robótica, deveriam apresentar o mesmo comportamento ao longo da simulação, uma vez que sua construção é idêntica. Mas, o comportamento ao longo da simulação apresentou leves diferenças.

O código construído cria uma tabela com 8 colunas, onde cada uma delas é responsável por armazenar os seguintes valores:

- Tempo (em segundos),
- Ângulo assumido pelo motor (em graus),
- Potência fornecida ao motor (em porcentagem),
- Velocidade linear assumida pelo motor (em metros por segundo),

- Velocidade linear assumida pelo motor (em milímetros por segundo),
- Velocidade angular assumida pelo motor (em rotações por minutos),
- Velocidade angular assumida pelo motor (em radianos por segundo)
- Velocidade angular assumida pelo motor (em graus por segundo).

Para cada servo motor grande foi realizado um conjunto de quatro testes, onde em cada um deles uma determinada porcentagem de potência era fornecida pelo bloco Ev3. Desse modo, utilizou-se os valores de porcentagem equivalentes a 25%, 50%, 75% e 100%.

Já para o servo motor médio foi realizado um conjunto de três testes, onde novamente em cada um deles uma determinada porcentagem de potência era fornecida pelo bloco Ev3. Desse modo, utilizou-se os valores de porcentagem equivalentes a 50%, 75% e 100%.

Foram utilizados apenas três valores de porcentagem de potência, pois valores inferiores a 50% não tinham a capacidade de movimentar o servo motor médio devido a relação de engrenagens.

Esses valores foram intercalados com momentos em que o bloco Ev3 não fornecia nenhuma porcentagem de potência ao motor, ou seja, o motor deveria sair do repouso, entrar em movimento e retornar ao repouso, assim sucessivamente.

Ao todo eram realizados um conjunto de 7 ciclos com motor em nível lógico baixo intercalados por 7 ciclos com o motor em nível lógico alto, estabelecendo a formação de uma onda retangular. A tensão elétrica fornecida pelo bloco Ev3 foi construída seguindo a lógica de modulação por largura de pulso (do inglês Pulse Widht Modulation), um sinal PWM.

Logo, o valor de tensão não foi avaliado em termos de Volts, mas sim em termos de porcentagem, pois era necessário estabelecer o tempo em que o motor estaria em nível lógico baixo e em nível lógico alto.

Em um primeiro momento, um intervalo de tempo equivalente a 10 milissegundos foi estabelecido para cada nível lógico. Esse intervalo, na prática se demonstrou extremamente pequeno, a ponto de o computador não conseguir coletar uma quantidade expressiva de dados do comportamento dos motores. O problema, estava relacionado com o tipo de sinal criado pelo computador em relação ao tipo de sinal criado pelo comportamento dos motores. Enquanto, o comportamento dos motores resultava na criação de um sinal analógico, o sinal criado pelo computador era um sinal digital.

Dessa maneira, eles eram de naturezas diferentes, e consequentemente a conversão de uma modalidade na outra não estava sendo feita de forma adequada. A solução encontrada para esse obstáculo foi estabelecer um número de amostras para cada conjunto de valores de porcentagem utilizados na simulação.

Cada um dos conjuntos deveria contar com 1000 amostras, totalizando 14000 amostras, divididas em dois conjuntos de 7000 amostras intercaladas. Ao final desse processo foram obtidas 19 tabelas:

- 4 para o servo motor grande direito com as rodas,
- 4 para o servo motor grande direito sem as rodas,
- 4 para o servo motor grande esquerdo com as rodas,
- 4 para o servo motor grande esquerdo com as rodas,
- 3 para o servo motor médio.

Posteriormente, cada uma das tabelas foi aberta no Microsoft Excel, onde passaram por uma conversão para o formato (.xlsx) suportado pelo recurso SIMULINK no Matlab. Para uma melhor organização desses dados coletados, eles foram separados adequadamente, evitando a ocorrência de conflitos no seu processamento para a obtenção do modelamento matemático.

A Figura (59) apresenta um trecho desse código responsável por realizar o tratamento de dados.

Figura 59 - Trecho do código responsável por realizar o tratamento dos dados.

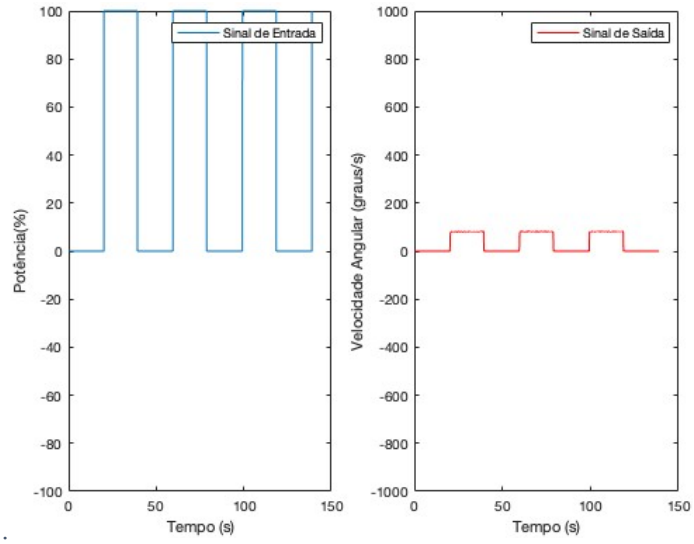
```
% Importação dos Dados Presentes no Arquivo (.xlsx)
% Carregamento dos dados referentes ao tempo de simulação:
time = xlsread("Dados_Motor_Direito_100%.xlsx", "A2:A14001");

% Carregameento dos dados referentes aos ângulos assumidos pelo motor:
angulo = xlsread("Dados_Motor_Direito_100%.xlsx", "B2:B14001");

% Carregamento dos dados referentes a potência fornecida ao motor:
potencia = xlsread("Dados_Motor_Direito_100%.xlsx", "C2:C14001");

% Carregamentos dos dados referentes a velocidade assumida pelo motor:
velocidadeL = xlsread("Dados_Motor_Direito_100%.xlsx", "D2:D14001");
velocidadeA = xlsread("Dados_Motor_Direito_100%.xlsx", "F2:F14001");
velocidadeR = xlsread("Dados_Motor_Direito_100%.xlsx", "H2:H14001");

% Inicialização:
pwm2voltage = 1; % Fator de Conversão entre PWM e Tensão elétrica

% Remove os dois pontos iniciais dos dados (Geralmente ruidoso):
time = time(3:end);
angulo = angulo(3:end);
potencia = potencia(3:end);
velocidadeL = velocidadeL(3:end);
velocidadeA = velocidadeA(3:end);
velocidadeR = velocidadeR(3:end);

% Cria um filtro passa-baixas IIR de segunda ordem:
d1 = designfilt('lowpassiir','DesignMethod','butter', ...
                'FilterOrder',2,'HalfPowerFrequency',3/25);

% Filtra os sinais adquiridos no DataLogging originados pelo Motor:
angulo = filter(d1,angulo);
velocidadeL = filter(d1,velocidadeL);
velocidadeA = filter(d1,velocidadeA);
velocidadeR = filter(d1,velocidadeR);
```

Fonte: Adaptado de Matlab (2016).

O código realiza uma separação das informações em dois grupos. Essa separação é importante, uma vez que o conjunto de dados coletados eram destinados a uma ferramenta chamada de “Parameter Estimator” presente no SIMULINK.

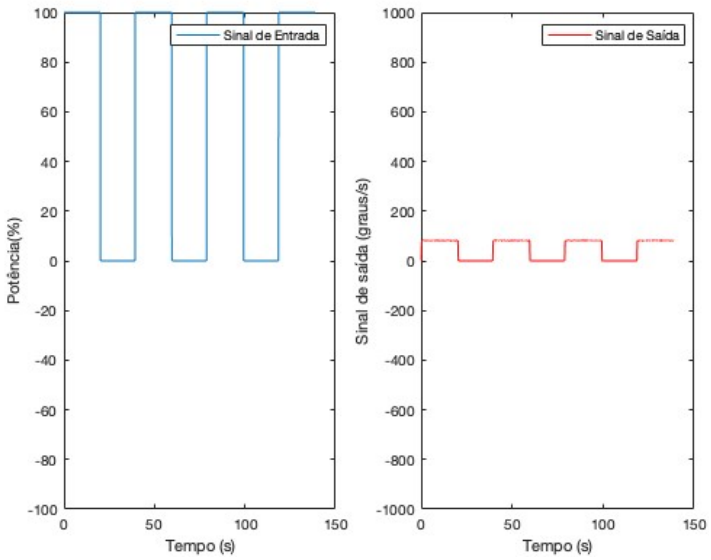
Os gráficos abaixo são a representação do comportamento do motor determinada pelo sistema de aquisição de dados.

Figura 60 – Comparação entre o sinal de entrada e o sinal de saída do motor destinados à estimação



Fonte: Autor

Figura 61 – Comparação entre o sinal de entrada e o sinal de saída do motor destinados à validação.



Fonte: Autor.

Essa ferramenta permite determinar as características associadas aos motores, ou seja, os valores dos parâmetros responsáveis por estabelecer o comportamento deles ao longo do funcionamento do robô.

3.1.7 Diagrama de blocos da equação diferencial de um motor de corrente contínua:

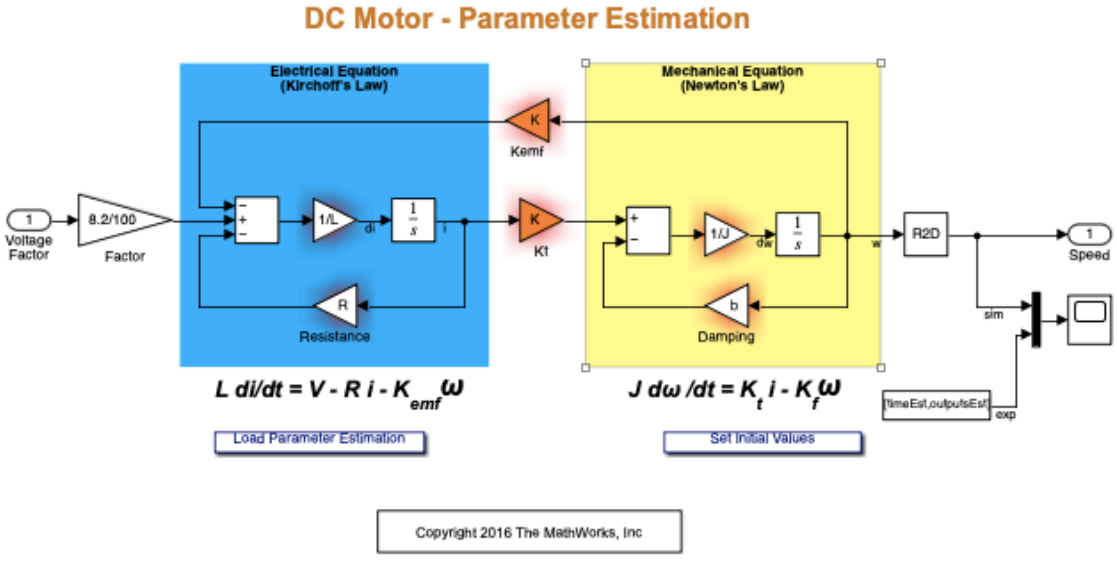
O uso do SIMULINK para a realização do modelamento matemático foi baseado nos documentos (MATHWORKS, 2013), (MATHWORKS, 2014) e (MATHWORKS, 2016).

Esses documentos foram produzidos pela MathWorks como um apanhado de informações introdutórias, ou seja, um workshop, tratando sobre o uso do LEGO® MINDSTORMS® em conjunto com o MatLab, assim como uma variedade de recursos.

Desse modo, eles foram utilizados como ponto de partida para o desenvolvimento dos modelos matemáticos. Porém, como o software sofreu várias alterações ao longo dos anos, alguns procedimentos foram adaptados as novas funcionalidades. Logo, existem algumas diferenças entre os processos conduzidos nas próximas seções e esses materiais.

O diagrama de blocos abaixo, representando uma equação diferencial de um motor de corrente contínua, foi o ponto chave para o desenvolvimento das próximas seções.

Figura 62 – Diagrama para determinação dos parâmetros de um motor elétrico de corrente contínua

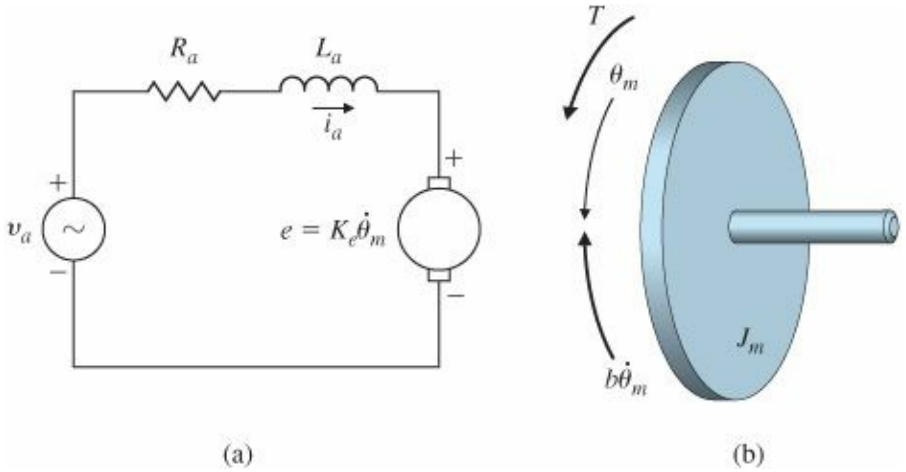


Fonte: MATHWORKS (2016).

3.1.8 Modelamento matemático dos servos motores:

O modelamento matemático dos servos motores foi conduzido com base na observação do diagrama de corpo livre apresentado abaixo (FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

Figura 63 – Diagrama de corpo livre da seção elétrica (a) e da seção mecânica (b).



Fonte: Adaptado de FRANKLIN; POWELL; EMAMI-NAEINI (2013-p.67).

Os servos motores empregados no protótipo são interpretados como motores elétricos de corrente contínua, ou seja, eles são sistemas eletromecânicos, onde a armadura representa a seção elétrica e o rotor representa a seção mecânica. Dessa forma, é possível separá-los em dois sistemas independentes, e posteriormente avaliá-los em conjunto (FRANKLIN; POWELL; EMAMI-NAEINI, 2013).

A armadura, sistema elétrico, é composta por uma fonte de tensão elétrica (tensão elétrica de entrada) representa pela variável $v_a\ (V)$, uma resistência elétrica de armadura representada pela variável $R_a\ (\Omega)$, uma indutância elétrica de armadura representada pela variável $L_a\ (H)$ e a força eletromotriz da armadura representada pela variável $E_a\ (V)$ (FRANKLIN; POWELL; EMAMI-NAEINI, 2013).

O rotor, sistema mecânico, é composto por uma massa (ou simplesmente inércia) representado pela variável $J_m\ (Kg.m^2)$, um amortecedor de atrito viscoso representado pela variável b (Ns/m^2) e o torque eletromagnético é representado pela variável $T\ (N.m)$ (FRANKLIN; POWELL; EMAMI-NAEINI, 2013).

Aplicando Lei de Kirchoff das Tensões em relação ao domínio do tempo obtém-se (FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

$$v_a(t) = e_a(t) + v_{R_a}(t) + v_{L_a}(t) \tag{10}$$

Cada uma das variáveis acima possui uma representação a partir de uma das equações características apresentadas abaixo (FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

$$v_a(t) = k_{fem}\omega(t) \tag{11}$$

$$v_{R_a}(t) = R_a i_a(t) \tag{12}$$

$$v_{L_a}(t) = L_a \frac{di_a(t)}{dt} \tag{13}$$

Onde a variável i_a $(Ampères)$ representa a corrente elétrica de armadura, a variável k_{fem} representa a constante elétrica associada a força eletromotriz da armadura. Substituindo as equações (11), (12) e (13) na equação (10) tem-se: (FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

$$v_a(t) = k_{fem}w(t) + R_a i_a(t) + L_a \frac{di_a(t)}{dt} \tag{14}$$

Aplicando Lei de Newton em relação ao domínio do tempo obtém-se (FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

$$T(t) = b(t) + J_{,}(t) \tag{15}$$

Cada uma das variáveis acima possui uma representação a partir de uma das equações características apresentadas abaixo (FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

$$T(t) = k_r i_a(t) \tag{16}$$

$$b_m(t) = b\omega(t) \tag{17}$$

$$J_m(t) = J\alpha(t) \tag{18}$$

Onde a variável i_r (A) representa a corrente elétrica do rotor, a variável k_m representa a constante mecânica associada ao torque eletromecânico, a variável b (Ns/m^2) representa a constante de amortecimento viscoso, a variável J $(Kg.m^2)$ representa o momento de inércia, a variável ω (rad/s) representa a velocidade angular e variável α (rad/s^2) representa a aceleração angular (FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

Substituindo as equações (16), (17) e (18) na equação (15), tem-se (FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

$$T(t) = bw(t) + J\alpha(t) \tag{19}$$

Ao interpretar a aceleração angular, como a primeira derivada da velocidade angular, a equação acima assume a seguinte forma (FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

$$k_r i_a(t) = bw(t) + J\frac{dw(t)}{dt} \tag{20}$$

Considerando as condições iniciais nulas, e aplicando a Transformada de Laplace nas equações (14) e (20), obtém-se (FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

$$V_a(s) = k_{fem}W(s) + R_a I_a(s) + sL_a I_a(s) \tag{21}$$

$$k_r I_a(s) = bW(s) + sJW(s) \tag{22}$$

Isolando o termo I_a nas equações (21) e (22) (FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

$$I_a(s) = \frac{V_a(s) - k_{fem}W(s)}{sL_a + R_a} \tag{23}$$

$$I_a(s) = \frac{bW(s) + sJW(s)}{k_r} \tag{24}$$

Igualando as equações (23) e (24) (FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

$$\frac{V_a(s) - k_{fem}W(s)}{sL_a + R_a} = \frac{bW(s) + sJW(s)}{k_r} \tag{25}$$

Como a função de transferência é uma relação entre a saída e a entrada do sistema, ou seja:

$$G(s) = \frac{Saída\ do\ Sistema}{Entrada\ do\ Sistema} = \frac{Y(s)}{X(s)} \tag{26}$$

Nessa situação, a entrada é correspondente a tensão elétrica de alimentação da armadura, e a saída é correspondente a velocidade angular. Logo, A equação (26) pode ser representada da seguinte forma (FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

$$\frac{V_a(s) - k_{fem}W(s)}{sL_a + R_a} = \frac{(b + Js)W(s)}{k_r} \tag{27}$$

$$V_a(s) = \frac{(b + Js)(sL_a + R_a)W(s)}{k_r} + k_{fem}W(s) \tag{28}$$

$$V_a(s) = \frac{\Big((b + Js)(sL_a + R_a)W(s) + k_{fem}k_rW(s)\Big)}{k_r} \tag{29}$$

$$V_a(s) = \frac{\Big((b + Js)(sL_a + R_a) + k_{fem}k_r\Big)W(s)}{k_r} \tag{30}$$

$$\frac{W(s)}{V_a(s)} = \frac{k_r}{(b + Js)(sL_a + R_a) + k_{fem}k_r} \tag{31}$$

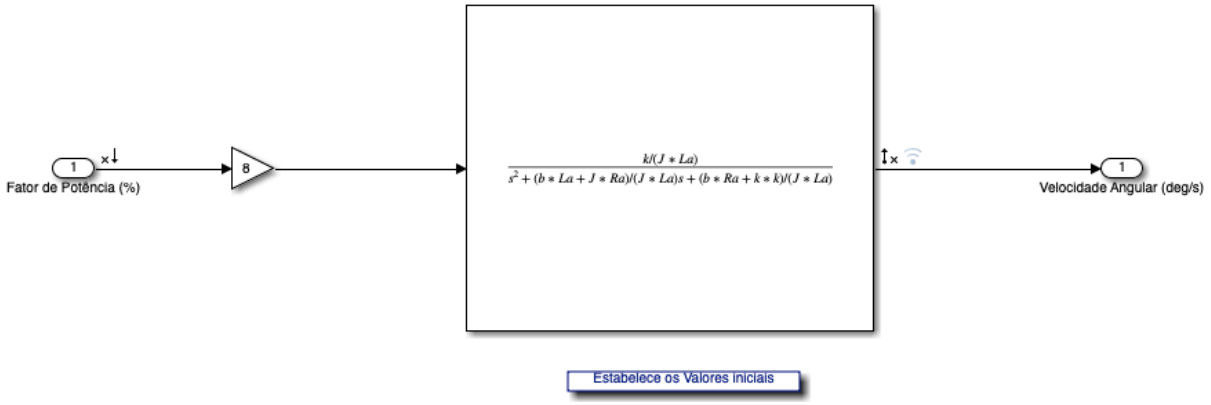
$$\frac{W(s)}{V_a(s)} = \frac{k_r}{JL_as^2 + JR_as + bL_as + bR_a + k_{fem}k_r} \tag{32}$$

$$\frac{W(s)}{V_a(s)} = \frac{k_r}{JL_as^2 + (bL_a + JR_a)s + \big(bR_a + k_{fem}k_r\big)} \tag{33}$$

$$\frac{W(s)}{V_a(s)} = \frac{\left(\frac{k_r}{JL_a}\right)}{s^2 + \frac{(bL_a + JR_a)}{JL_a}s + \frac{\big(bR_a + k_{fem}k_r\big)}{JL_a}} \tag{34}$$

A partir da função de transferência obtida no ambiente virtual do Matlab, um código foi elaborado com o objetivo de importar os dados presentes nas 19 tabelas referentes às combinações de potências dos motores. Um diagrama de blocos contendo a função de transferência acima foi construído, conforme apresentado pela Figura 64.

Figura 64 – Diagrama de blocos da Função de transferência adaptado para obtenção dos parâmetros do motor.



Fonte: Autor.

A simulação desse diagrama de blocos é fundamentada sobre a execução de dois códigos. O primeiro, já apresentado e comentado anteriormente, transfere os dados contidos em cada uma das colunas das tabelas, criando variáveis responsáveis por estabelecer (MATHWORKS, 2013; MATHWORKS, 2014):

- a passagem de tempo (variável tempo),
- o valor de tensão fornecido em termos de porcentagem (variável potencia)
- a velocidade angular assumida pelo motor em análise (variáveis velocidadeA, velocidade angular em RPM, e velocidadeR, velocidade angular em rad/s).

Os dados atribuídos são submetidos a uma filtragem para reduzir o ruído presente nas amostras, como também são separados em dois grupos (MATHWORKS, 2013; MATHWORKS, 2014).

Essa separação em dois grupos é exigida, para que seja possível estimar e validar os parâmetros encontrados, permitindo verificar se o comportamento virtual, era de fato semelhante ao comportamento real (MATHWORKS, 2013; MATHWORKS, 2014).

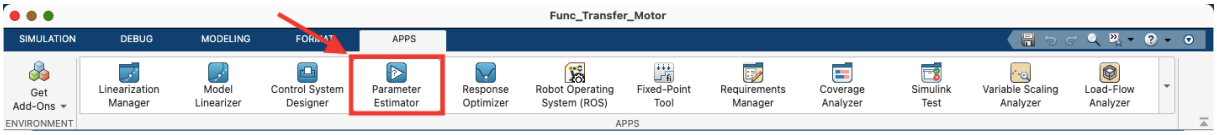
Já o segundo código estabelece os valores iniciais dos parâmetros b, J, La, kr, kfem e Ra, isto é, cria variáveis de mesmo nome (FRANKLIN; POWELL; EMAMI-NAEINI, 2013).

Vale ressaltar que na simulação foi conduzida uma simplificação igualando a constante mecânica associada ao torque eletromecânico (kr) e a constante elétrica associada a força eletromotriz da armadura (kfem), uma vez que possuem valores pequenos, e praticamente idênticos. Portanto, a função de transferência utilizada no programa é (FRANKLIN; POWELL; EMAMI-NAEINI, 2013):

$$\frac{\left(\frac{k}{JL_a}\right)}{s^2 + \frac{(bL_a + JR_a)}{JL_a}s + \frac{(bR_a + k^2)}{JL_a}} \tag{35}$$

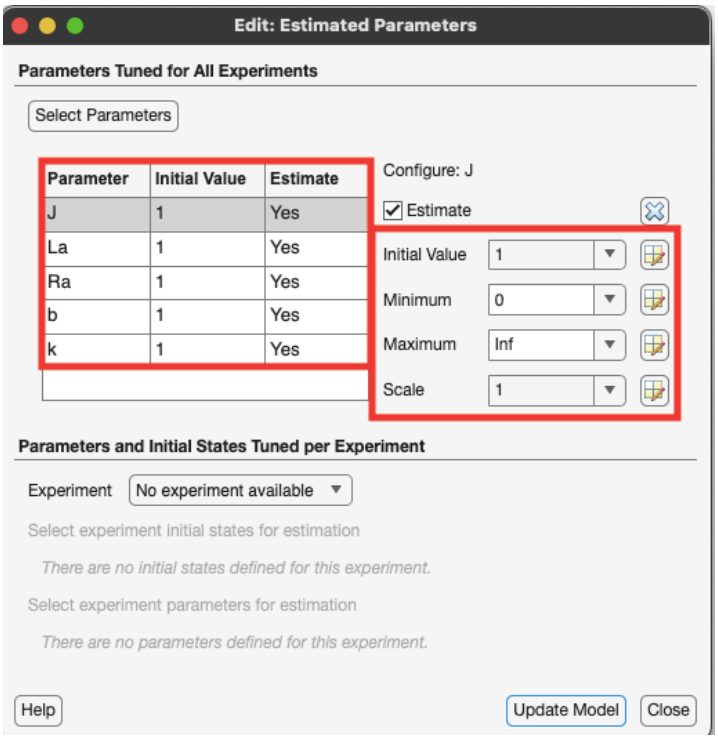
Com os dois códigos e o diagrama prontos, a ferramenta “Parameter Estimator” foi inicializada. A inicialização consiste na configuração das variáveis a serem estimadas. Nas figuras abaixo são destacadas cada uma das telas de configuração:

Figura 65 – Ícone do aplicativo "Parameter Estimator" na aba "APPS" do Simulink.



Fonte: Autor.

Figura 66 – Configuração das características básicas associadas a cada um dos parâmetros a serem estimados.

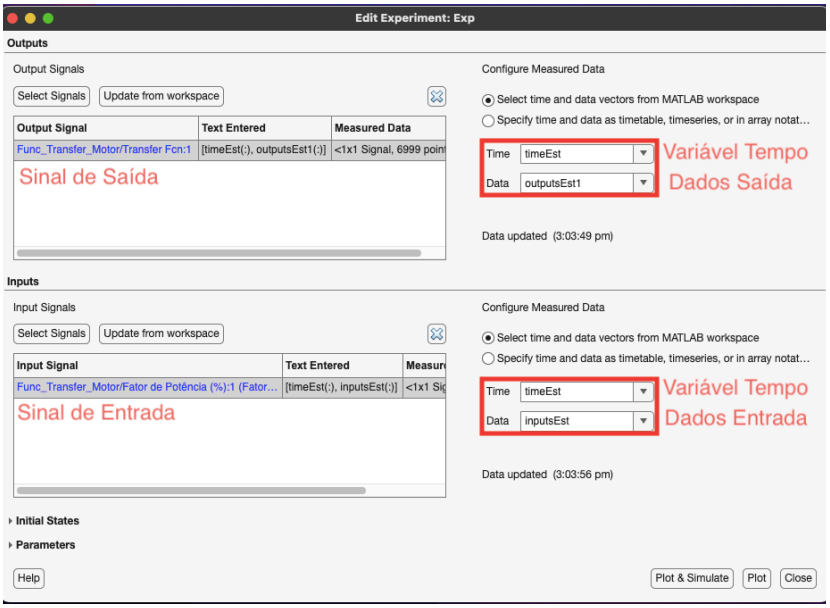


Fonte: Autor.

Os parâmetros b, J, La, k e Ra foram dimensionados considerando que seu valor está compreendido em um intervalo com limite inferior equivalente a 0 e o limite superior equivalente a infinito. A escala da representação gráfica nas janelas de simulação para estimação e para validação estão em uma proporção 1:1.

A Figura 67 ilustra a tela referente a configuração do experimento de estimação dos parâmetros. Foram selecionadas as variáveis inputsEst, outputsEst1 e timeEst, relacionadas aos dados de entrada, aos dados de saída e às informações da passagem de tempo, respectivamente.

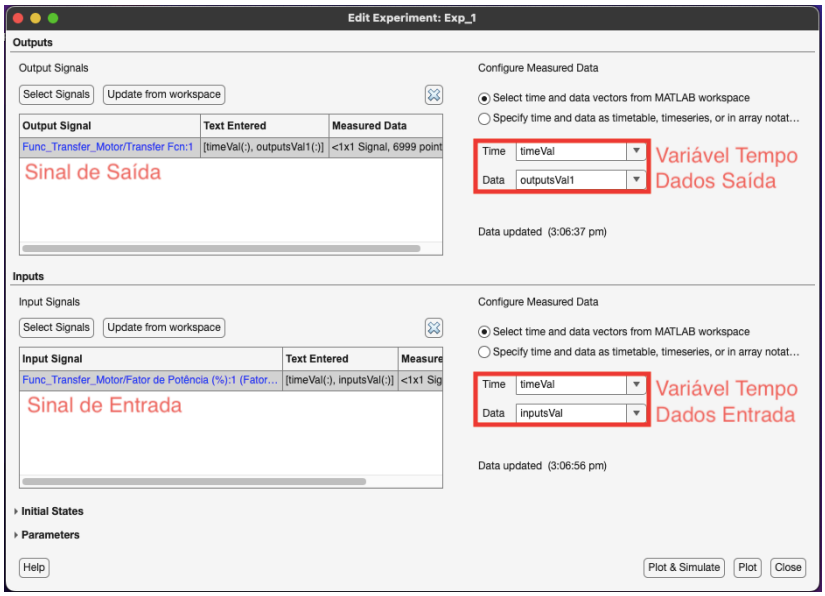
Figura 67 – Tela de configuração do experimento destinado a estimação.



Fonte: Autor.

A Figura 68 ilustra a tela referente a configuração do experimento de validação dos parâmetros. Foram selecionadas as variáveis inputsVal, outputsVal1 e timeVal, relacionadas aos dados de entrada, aos dados de saída e às informações do tempo, respectivamente.

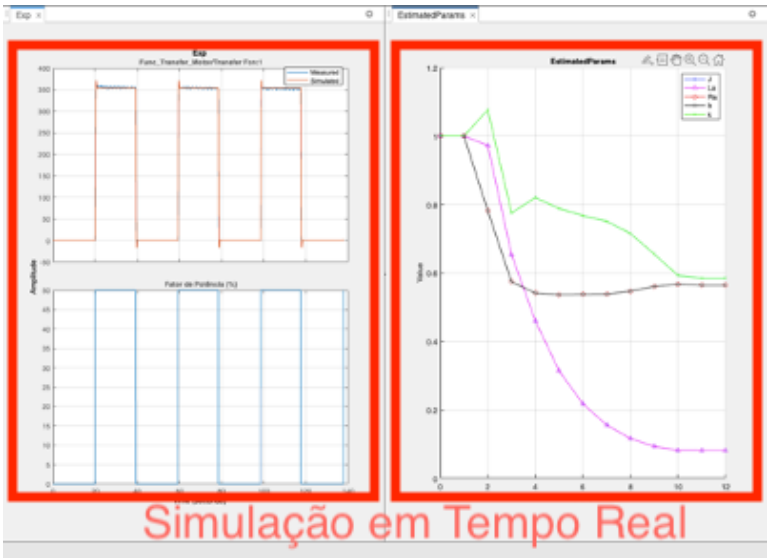
Figura 68 – Tela de configuração do experimento destinado a validação.



Fonte: Autor.

Após conduzir esses procedimentos o resultado é apresentado pela Figura 69 e pela Figura 70.

Figura 69 – Experimento utilizado na estimação (à esquerda) e processo de determinação dos valores associados a cada parâmetro (à direita).



Fonte: Autor.

Figura 70 – Comparação entre o experimento dedicado à estimação dos parâmetros (à esquerda) e do experimento dedicado à validação dos parâmetros (à direita).

Fonte: Autor.

Ao final do procedimento, a ferramenta retornava o valor assumido pelos parâmetros, como ilustrado pela Figura 71:

Figura 71 – Exemplo de resultado obtido para a simulação.

View Result: EstimatedParams

```
Estimation result(s):
    J = 0.081766
    La = 0.081766
    Ra = 0.56403
    b = 0.56403
    k = 0.58497
```

Fonte: Autor

Como uma grande variedade de testes foi executada, foram criadas as Tabelas 6, 7, 8, 9 e 10 para assegurar uma melhor organização dos dados. Posteriormente, as funções de transferência associadas a cada conjunto foram organizadas nas Tabelas 11, 12, 13, 14 e 15.

Nos experimentos feitos com as rodas acopladas ao motor direito e ao motor esquerdo, foi verificada a existência de alguns valores distintos daqueles encontrados nos experimentos sem as rodas acopladas. Para certas potências, os valores acabaram ficando um pouco acima do esperado.

Essa variação muito provavelmente está associada a estrutura das rodas que é composta por uma relação de engrenagens e eixos, que acabaram repercutindo em um comportamento diferente para certos valores de porcentagem de tensão elétrica.

Como consequência, foi constatada uma flutuação de alguns dos parâmetros dos motores, que estão em destaque nas tabelas seguintes com a coloração laranja. Para evitar a ocorrência de mais erros no cálculo dos ganhos presentes nos controladores, as funções de transferência, também destacadas com a coloração laranja, foram desconsideradas nas próximas etapas.

Os valores na coloração laranja foram desconsiderados, uma vez que ao longo dos testes realizados, foi possível verificar que existia uma tendência no comportamento dos parâmetros relacionados aos motores.

Mais precisamente, os valores que garantiam uma maior proximidade do comportamento real eram aqueles em que o valor de b era idêntico ao valor de Ra, assim como o valor de J era idêntico ao valor La.

Tabela 6 – Motor direito sem as rodas.

Tensão Elétrica (%)	b (Ns/m^2)	J ($Kg.m^2$)	K	La (H)	Ra (Ω)
25	0.48715	0.067523	0.49143	0.067523	0.48715
50	0.45188	0.06337	0.45102	0.06337	0.45188
75	0.46096	0.065575	0.4521	0.065575	0.46096
100	0.45827	0.066444	0.41247	0.066444	0.45827

Fonte: Autor.

Tabela 7 – Motor direito com as rodas.

Tensão Elétrica (%)	b (Ns/m²)	J (Kg.m²)	K	La(H)	Ra (Ω)
25	0.35244	0.35636	8.0148	0.46784	27.543
50	0.87278	0.1193	0.83527	0.1193	0.87278
75	0.84011	0.12342	0.83748	0.12342	0.84011
100	0.81514	0.11463	0.74598	0.11463	0.81514

Fonte: Autor.

Tabela 8 – Motor esquerdo sem as rodas.

Tensão Elétrica (%)	b (Ns/m²)	J (Kg.m²)	K	La (H)	Ra (Ω)
25	0.61509	0.89273	0.64199	0.089273	0.61509
50	0.56403	0.081766	0.58497	0.081766	0.56403
75	0.53333	0.075718	0.51585	0.075718	0.53333
100	0.50886	0.069281	0.45089	0.068281	0.50886

Fonte: Autor.

Tabela 9 – Motor esquerdo com as rodas.

Tensão Elétrica (%)	b (N.m.s)	J (Kg.m²)	K	La (H)	Ra (Ω)
25	1.5189	0.50663	5.2407	0.49034	10.602
50	1.8203	0.061737	2.2576	0.65045	2.3065
75	2.6953	0.076706	2.6547	0.59784	1.9805
100	0.8075	0.11353	0.76097	0.11353	0.8075

Fonte: Autor.

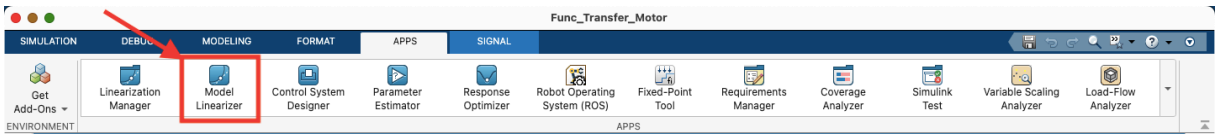
Tabela 10 – Motor médio com plataforma elevadora.

Tensão Elétrica (%)	b (N.m.s)	J (Kg.m²)	K	La (H)	Ra (Ω)
50	1.0941	0.12886	1.0675	0.12886	1.0941
75	2.1085	0.3155	4.3938	0.42994	13.439
100	1.7877	0.062492	3.1305	0.72361	0.93582

Fonte: Autor

As funções de transferência foram calculadas virtualmente. Esse procedimento exigiu o uso de outra ferramenta presente no SIMULINK, o “Model Linearizer”.

Figura 72 – Ícone do aplicativo "Model Linearizer" na aba "APPS" do Simulink.

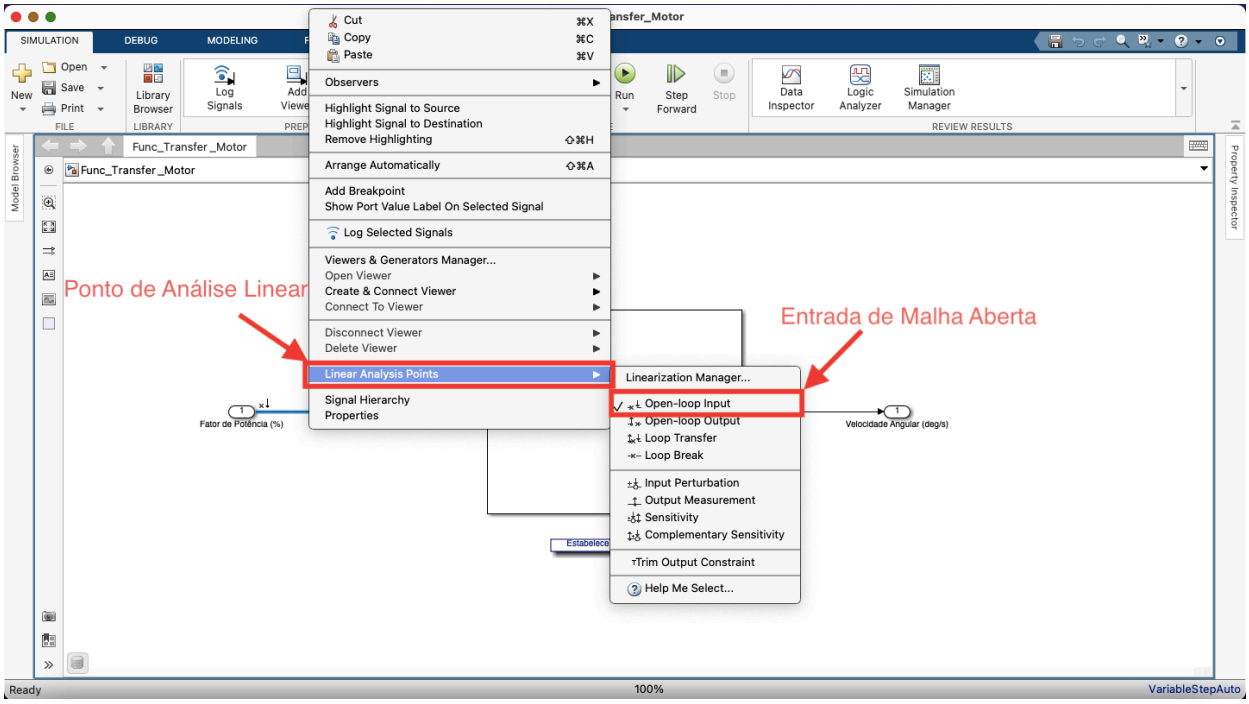


Fonte: Autor.

A utilização do “Model Linearizer” depende do diagrama de blocos criado para o modelamento do motor, uma vez que seu uso leva em consideração a quantidade de entradas e saídas do sistema.

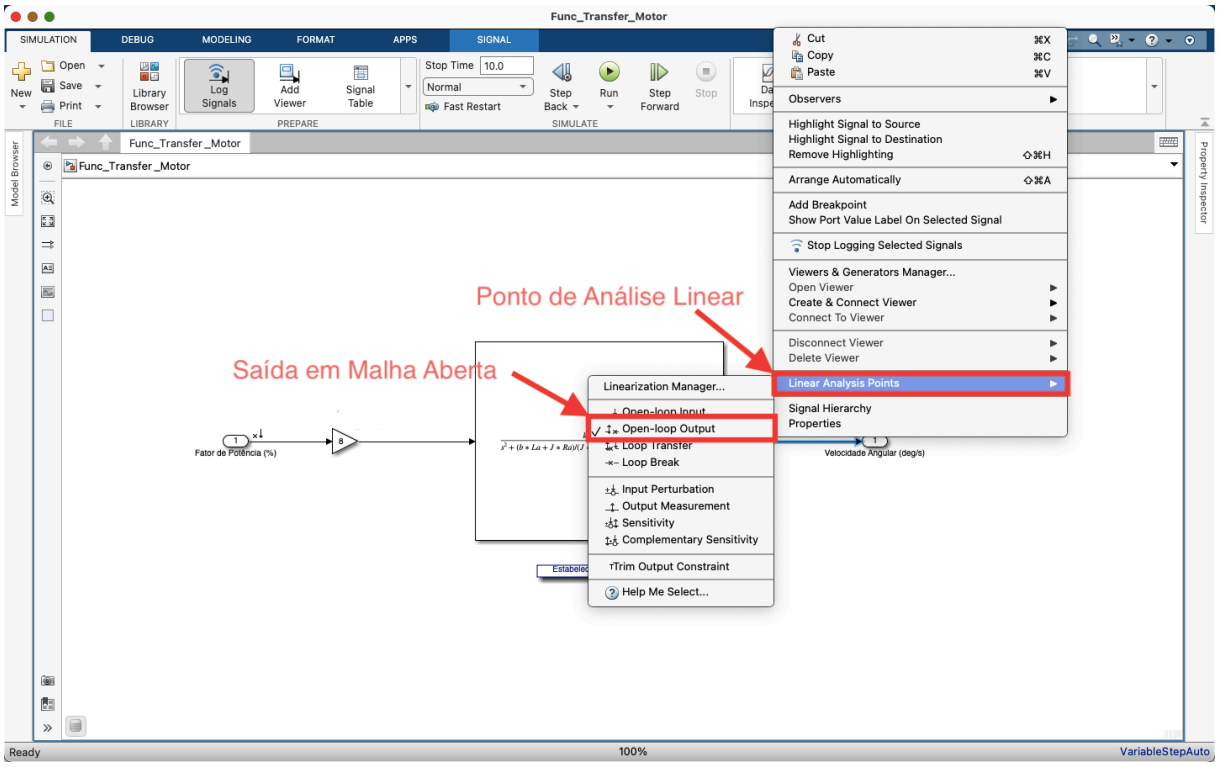
O sistema, em questão, apresenta uma entrada e uma saída, que foram devidamente sinalizadas a partir das marcações indicadas na Figura 73 e na Figura 74:

Figura 73 – Definição do ponto de análise linear para a entrada do sistema.



Fonte: Autor.

Figura 74 – Definição do ponto de análise linear para a saída do sistema.



Fonte: Autor.

Após a configuração da entrada e da saída, foi feita a simulação para obtenção da função de transferência ligada a cada conjunto de valores. As tabelas abaixo apresentam a estrutura das funções de transferência encontradas para os motores sem rodas e com rodas:

Tabela 11 – Função de transferência motor direito sem rodas

Tensão Elétrica (%)	Função de Transferência G(s)
25	$\frac{862.3}{s^2 + 14.43s + 105}$
50	$\frac{898.5}{s^2 + 14.26s + 101.5}$
75	$\frac{841.1}{s^2 + 14.06s + 96.95}$
100	$\frac{747.4}{s^2 + 13.79s + 86.1}$

Fonte: Autor.

Tabela 12 – Função de transferência motor direito com rodas

Tensão Elétrica (%)	Função de Transferência G(s)
25	$\frac{384.6}{s^2 + 13.1s + 97.1}$
50	$\frac{469.5}{s^2 + 14.63s + 102.5}$
75	$\frac{439.9}{s^2 + 13.61s + 92.38}$
100	$\frac{454.2}{s^2 + 14.22s + 92.92}$

Fonte: Autor.

Tabela 13 – Função de transferência motor esquerdo sem rodas

Tensão Elétrica (%)	Função de Transferência G(s)
25	$\frac{644.4}{s^2 + 13.78s + 99.19}$
50	$\frac{700}{s^2 + 13.8s + 98.77}$
75	$\frac{719.8}{s^2 + 14.09s + 96.03}$
100	$\frac{751.5}{s^2 + 14.69s + 96.3}$

Fonte: Autor.

Tabela 14 – Função de transferência motor esquerdo com rodas

Tensão Elétrica (%)	Função de Transferência G(s)
25	$\frac{168.8}{s^2 + 5.919s + 42.17}$
50	$\frac{449.8}{s^2 + 13.97s + 971}$
75	$\frac{463.1}{s^2 + 13.74s + 96.54}$
100	$\frac{472.2}{s^2 + 14.22s + 95.49}$

Fonte: Autor.

Tabela 15 – Função de transferência motor médio com plataforma elevadora

Tensão Elétrica (%)	Função de Transferência G(s)
50	$\frac{514.3}{s^2 + 16.98s + 140.7}$
75	$\frac{309.1}{s^2 + 8.804s + 77.32}$
100	$\frac{553.8}{s^2 + 15.66s + 132.8}$

Fonte: Autor.

As funções de transferência destacadas na coloração laranja acabaram descartadas. Como existiam discrepâncias nos parâmetros empregados para determiná-las, existia uma grande chance do erro associado a elas se propagar para os demais passos, consequentemente afetando o desempenho das próximas simulações.

3.1.9 Modelamento do Sensor de Seguimento de Linha e Controlador ON/OFF:

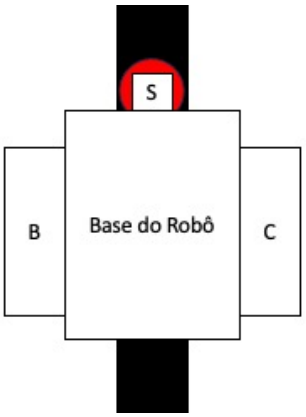
Conforme apresentado na seção 2.7.3, os sensores utilizados para seguir uma linha possuem o seu funcionamento baseado na emissão de luz. Em muitas situações eles também são chamados de sensores de luz, uma vez que seu funcionamento está de acado com a ativação do LED vermelho presente em sua estrutura.

Esse LED vermelho possui uma variedade de comportamentos, contudo o mais básico e imprescindível para esse experimento é sua capacidade de medir a intensidade luminosa da luz refletida. A luz vermelha é emitida e por meio de sua reflexão é possível determinar a coloração da superfície dentro de um espaço limitado de valores.

Conforme demonstrado na seção 3.1.2, o robô possui dois motores (B e C) dedicados a realizar o seu deslocamento, onde cada um deles está conectado a uma das duas esteiras disponíveis.

Já o sensor de luz, foi posicionado na parte dianteira com uma pequena altura em relação ao solo, impedindo que ele tenha contato com outra região do ambiente que não seja a superfície da base do trajeto.

Figura 75 – Diagrama representando o robô posicionado sobre o trajeto.



Fonte: Adaptado de Inpharmix (2018).

No esquema acima, é possível identificar cada um desses elementos, com destaque para a região em vermelho. Essa região indica a área compreendida pelo processo de detecção do sensor, ou seja, somente no interior dela os dados e as informações do ambiente são identificados. Já o restante, no exterior, é desconsiderado.

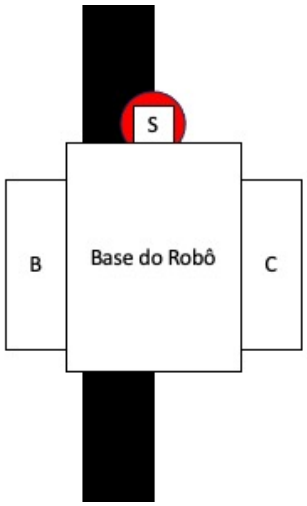
O intuito é fazer o robô se deslocar ao longo da linha preta, isto é, a fita isolante. Sendo assim, ele deve evitar se deslocar sobre a região branca, isto é, o ambiente ao redor dela. Porém, o uso de um sensor de luz acaba acarretando algumas limitações no movimento, principalmente na identificação dessas duas regiões.

Devido a isso, ao invés de se deslocar sobre a linha preta, o robô deverá se deslocar ao longo da borda, pois ao fazê-lo se deslocar pelo centro da linha preta existe um problema de referência.

Esse problema é derivado do fato do trajeto apresentar duas bordas, a esquerda e a direita. Dessa maneira, isso cria uma espécie de ambiguidade, fazendo com que o movimento seja impreciso.

Logo, a referência deve ser uma das bordas da linha disponíveis, visto que a partir do momento em que o sensor realizar uma leitura correspondente a cor branca é possível inferir se ele está à direita da borda de referência, ou a esquerda dela. As figuras abaixo demonstram uma visão simplificada dessa situação.

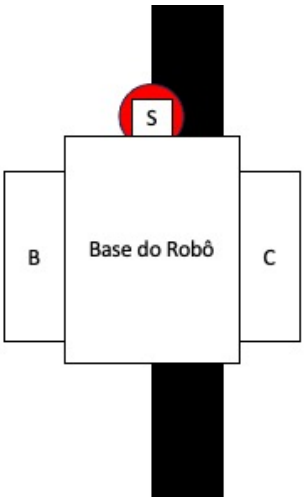
Figura 76 – Diagrama de seguidor de linha posicionado sobre a borda direita.



Fonte: Adaptado de Inpharmix (2018).

Na Figura 76, caso o sensor compute uma leitura da coloração preta, ele está à direita da borda, enquanto caso o sensor compute uma leitura da coloração branca, ele está à esquerda da borda.

Figura 77 – Diagrama de seguidor de linha posicionado sobre a borda esquerda.



Fonte: Adaptado de Inpharmix (2018).

Na figura 77, caso o sensor compute uma leitura da coloração preta, ele está à esquerda da borda, enquanto caso o sensor compute uma leitura da coloração branca, ele está à direita da borda.

Para o funcionamento adequado do sensor de luz na forma de um sensor de seguimento de linha, inicialmente foi preciso compreender sua capacidade de detecção da intensidade luminosa sobre a superfície da base utilizada, como também da fita isolante preta empregada.

Esse procedimento foi feito a partir da elaboração de um código simples desenvolvido no Visual Studio Code, com o uso da biblioteca MicroPython e as extensões fornecidas pela ev3dev.

Basicamente, ao conectar o sensor ao bloco Ev3 e posicioná-lo sobre uma superfície, ele retornava um valor correspondente à porcentagem da coloração, sendo 0% mais escuro e 100% mais claro. Abaixo é apresentado o código construído para executar essa ação:

Figura 78 – Código responsável por realizar a verificação da intensidade luminosa das superfícies.

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# Cria o objeto relacionado ao bloco Ev3
ev3 = EV3Brick()

# Cria o objeto relacionado ao sensor de cor e associa a porta utilizada no bloco Ev3
luz = ColorSensor(Port.S1)

# Simulação do Programa:
# Cada 10 milissegundos uma leitura é feita para computar a porcentagem da intensidade
# da superfície onde o sensor foi posicionado.
while True:
    # Criação da variável porcentagem para armazenar o valores de cada leitura.
    porcentagem = luz.reflection()
    # Impressão da leitura atual na tela do terminal do computador.
    print(porcentagem)
    # Espera 10 milissegundos para realizar uma nova leiitura.
    wait(10)
```

Fonte: Adaptado de The Cooding Fun (2020).

De acordo com as leituras conduzidas por meio do sensor, a coloração preta foi computada na faixa de 5% e a coloração branca foi computada na faixa de 50%. A determinação desses dois valores é extremamente importante, pois permite encontrar o chamado limiar de transição.

O limiar de transição é a média aritmética dos dois valores extremos computados pelo sensor, logo seu valor é equivalente a 27,5%. Essa informação, em conjunto com a posição específica do robô em relação a borda da linha, permite estabelecer qual intervalo de valores é destinado a uma curva à direita ou a uma curva à esquerda.

Figura 79 – Diagrama da leitura do sensor em função da direção da curva.

Fonte: Fonte: Adaptado de Inpharmix (2018).

Na Figura 79, caso o valor seja maior que 27,5%, será feita uma curva à direita, enquanto, caso o valor seja menor que 27,5%, será feita uma curva à esquerda, e assim sucessivamente ao longo de todo o trajeto.

Esse seguimento de linha utiliza a lógica presente em um controlador do tipo ON-OFF. Abaixo uma imagem é apresentada para mostrar a construção dessa lógica no código responsável por implementar o controlador no robô. O código completo está presente no Apêndice D.

Figura 80 – Lógica empregada na implementação do controlador ON/OFF.

```
38  # Inicializa a avaliação do comportamento do robô.
39  watch = StopWatch()
40
41  while True:
42      # Realiza a leitura da intensidade luminosa conduzida pelo sensor.
43      leitura = sluz.reflection()
44      # Constroí uma lista com os valores de cada leitura conduzida pela sensor.
45      vleitura.append(leitura)
46
47      # Calcula o valor do erro subtraindo a leitura atual do limiar.
48      erro = sluz.reflection() - limiar
49      # Constroí uma lista com os valores de cada erro calculado.
50      verro.append(erro)
51
52      # Caso a leitura seja menor ou igual ao limiar, o robô vira à esquerda
53      if leitura <= limiar:
54          mesquerdo.run_angle(DRIVE_SPEED,50)
55          mdireito.stop()
56
57      # Caso contrário, o robô vira à direita
58      else:
59          mdireito.run_angle(DRIVE_SPEED,50)
60          mesquerdo.stop()
61
62      # Estabelece a avaliação do tempo decorrido durante a simulação.
63      T = watch.time()
64      T = T*0.001
65
66      # Computa os valores do tempo, do erro e da curva em uma linha da tabela de dados.
67      data.log(T, erro)
68      # Espera 10 milissegundos para a realização da próxima leitura.
69      wait(10)
```

Fonte: Adaptado de The Cooding Fun (2020).

3.1.10 Controlador 3 Níveis:

O controlador ON-OFF tem uma lógica fácil e um funcionamento bastante simples, contudo seu desempenho possuí algumas variações em trajetos onde existem curvas mais acentuadas.

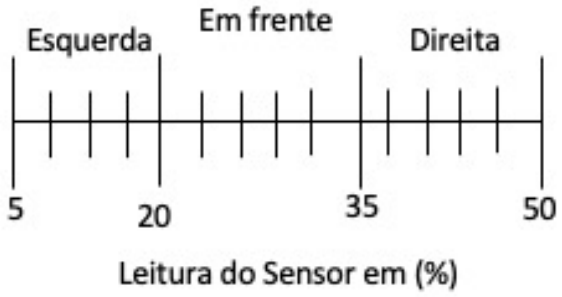
Curvas mais acentuadas exigem uma combinação precisa de valores relacionadas a potência fornecida pelo bloco Ev3 aos motores. Outro fator exigido é um conhecimento de qual das rodas está sendo fundamental para a realização do deslocamento a cada instante, para que seja possível estabelecer velocidades corretas. Em certos casos, até mesmo uma parada abrupta de um dos motores é necessária.

Desse modo, o controlador precisa de uma lógica mais robusta para atender percursos mais complexos, uma vez que ele apenas assimila informações capazes de movimentá-lo para esquerda ou para a direita.

Logo, não é uma abordagem muita precisa, ou até mesmo muito veloz caso o objetivo seja fazer um percurso sem grandes erros dentro de um tempo estabelecido. A primeira possibilidade de resolução desse problema é divisão da faixa de valores compreendida entre os dois extremos em mais uma fração, ou seja, totalizando três intervalos.

Além dos intervalos responsáveis pelas ações de virar à direita e de virar à esquerda, entre eles um novo conjunto de valores seria estabelecido para permitir o movimento em linha reta.

Figura 81 – Diagrama da leitura do sensor em função de diferentes níveis.



Fonte: Fonte: Adaptado de Inpharmix (2018).

.

Na figura acima, caso o valor seja maior que 5% e menor que 20%, será feita uma curva à esquerda. Já, caso o valor seja maior que 20% e menor que 35%, seguirá em frente. Por último, caso o valor seja maior que 35% e menor que 50%, será feita uma curva a direita, e assim sucessivamente ao longo de todo o trajeto.

Esse seguimento de linha utiliza a base de um controlador ON-OFF, porém a nova divisão permite mais uma categoria de movimento. Mesmo sendo mais robusto, fazendo com que o robô desempenhe um movimento retilíneo em determinados momentos, ele ainda não é capaz de lidar com certas situações.

No entanto, não é possível negar que a presença de uma nova divisão compreendida entre os dois extremos garantiu um movimento do robô seguidor de linha mais próximo do ideal. Ou seja, a adição de mais divisões ao intervalo original possibilita ao robô realizar uma maior variedade de curvas à direita ou à esquerda. Como resultando diferentes maneiras de realizar uma curva são possíveis.

A seguir uma imagem ilustra a construção dessa lógica no código responsável por implementar o controlador no robô. O código completo está presente na seção Apêndice E.

Figura 82 – Lógica empregada na implementação do controlador de 3 níveis.

```
# Calcula o limiar baseado nos valores atribuídos as cores branca e preta.
preto = 5
branco = 50
limiar1 = int((branco - preto/3)+preto
limiar2 = branco - int((branco - preto)/3)

# Estabelece a velocidade em 200 milimetros por segundo.
DRIVE_SPEED = 200

# Inicializa a avaliação do comportamento do robô.
watch = StopWatch()

while True:
    # Realiza a leitura da intensidade luminosa conduzida pelo sensor.
    leitura = sluz.reflection()
    # Constroí uma lista com os valores de cada leitura conduzida pela sensor.
    vleitura.append(leitura)

    # Caso a leitura seja menor ou igual ao limiar1, o robô fará uma curva à direita.
    if leitura <= limiar1:
        mdireito.run_angle(DRIVE_SPEED,50)
        mesquerdo.stop()
        erro = sluz.reflection() - limiar1
        verro.append(erro)

    # Caso a leitura seja maior ou igual ao limiar2, o robô fará uma curva à esquerda.
    elif leitura >= limiar2:
        mesquerdo.run_angle(DRIVE_SPEED,50)
        mdireito.stop()
        erro = sluz.reflection() - limiar2
        verro.append(erro)

    # Caso contrário o robô seguirá em frente
    else:
        mdireito.run(DRIVE_SPEED)
        mesquerdo.run(DRIVE_SPEED)
        erro = 0
        verro.append(erro)
```

Fonte: Adaptado de The Cooding Fun (2020).

3.1.11 Controlador Proporcionl:

A partir do princípio lógico responsável por estabelecer o controlador de 3 Nívies, é possível compreender que uma maneira interessante de avaliar esse intervalo seria considerá-lo como contendo infinitos números.

As principais referências estariam associadas aos pontos extremos, indicando a necessidade de desenvolver uma curva para uma das duas direções possíveis, e um ponto central, indicando a necessidade de realizar um movimento em linha reta.

Sendo assim, o que passa a ser avaliado não é mais apenas o valor médio, mas a diferença existente entre a leitura e o ponto central de referência. Essa diferença, ou taxa de variação, é utilizada como parâmetro para avaliar se a curva será mais aberta ou mais fechada.

Convertendo para os elementos do robô, essas grandezas corresponderiam ao erro associado à curva e a capacidade de giro ao longo da curva. Inicialmente, a proporcionalidade parte de um conceito bastante simples, correlacionado as duas grandezas em análise.

Entretanto, o mais importante dessa relação é o fato de que ela pode ser representada na forma de uma reta, isto é, uma função do primeiro-grau. Como consequência sua representação é definida pela forma:

$$f(x) = ax + b \tag{36}$$

Onde o termo f(x) é a variável dependente, x é a variável independente, a é o coeficiente angular e corresponde a inclinação da reta, b é o coeficiente linear e corresponde ao ponto onde a reta cruza o eixo y.

Dessa maneira, o centro da reta deve ser equivalente a 27,5%, isto é, o valor do limiar calculado anteriormente. Essa mudança é exigida, pois ele funciona como a base de cálculo do erro existente entre um dos pontos extremos e o valor computado pela leitura do sensor. Essa lógica deixa mais simples determinar se o robô está próximo da linha, ou distante dela.

Feita a simplificação, o sistema de coordenadas cartesianas permite o cálculo mais rápido dos erros em relação a nova leitura feita pelo sistema. Assim, a função correspondente ao controlador proporcional apresenta a seguinte estrutura:

$$f(x) = ax + b \rightarrow Como\ cruza\ na\ origem\ do\ eixo\ cartesiano \tag{37}$$
$$isto\ \text{é}, O = (0,0) \therefore b = 0$$

Como resultado a equação (36) assume a forma:

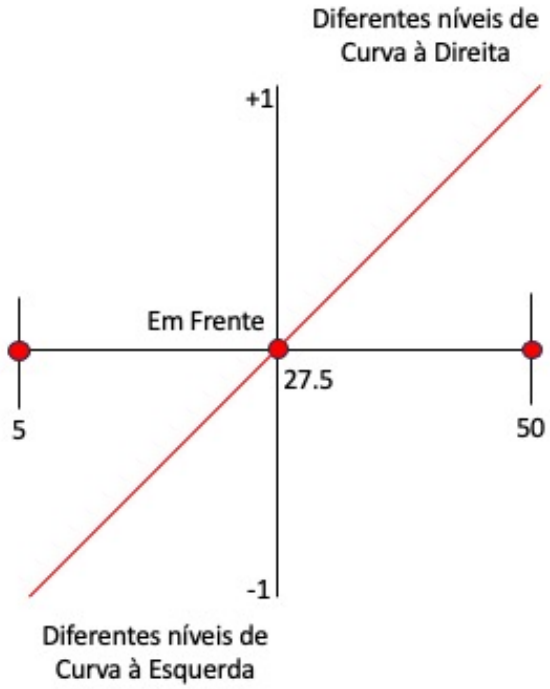
$$f(x) = a \tag{38}$$

O coeficiente angular (a) nessa relação se manifesta como o ganho proporcional K, fazendo jus ao nome do controlador.

Essa relação somente é válida para as curvas, visto que uma leitura equivalente a 27,5% não proporcionará um erro propriamente dito. Isso decorre do fato do valor em questão ser a referência para um movimento em linha reta. O deslocamento para direita e para esquerda é feito, quando um valor positivo, ou negativo é assumido.

O plano cartesiano presente na imagem abaixo representa o comportamento do robô de acordo com a função do primeiro grau desenvolvida nas equações acima:

Figura 83 – Comportamento do robô de acordo com a função característica do controlador P.



Fonte: Fonte: Adaptado de Inpharmix (2018).

.

Como o deslocamento depende da porcentagem de tensão elétrica aplicada aos dois motores simultaneamente, a melhor solução é acrescentar um termo para readequar a função. Essa readequação é feita sobre o código desenvolvido para a implementação do controlador.

As duas equações construídas abaixo destacam o encadeamento lógico entre os processos implementados no código para os motores funcionarem adequadamente:

$$erro = (leitura\ do\ sensor) - (limiar) \tag{39}$$

$$Curva = (ganho\ proporcional) \cdot (erro) \tag{40}$$

$$Alimentação\ do\ motor = (Velocidade\ de\ Deslocamento) + (Curva) \quad (41)$$

A imagem abaixo ilustra o trecho do código desenvolvido nesse procedimento. O código completo para a implementação desse controlador está presente na seção Apêndice (F).

Figura 84 – Lógica empregada na implementação do controlador P.

```
37  PROPORTIONAL_GAIN = -5
38  # Inicializa a avaliação do comportamento do robô.
39  watch = StopWatch()
40
41  while True:
42      # Realiza a leitura da intensidade luminosa conduzida pelo sensor.
43      leitura = sluz.reflection()
44      # Constroí uma lista com os valores de cada leitura conduzida pela sensor.
45      vleitura.append(leitura)
46
47      # Calcula o valor do erro subtraindo a leitura atual do limiar.
48      erro = sluz.reflection() - limiar
49      # Constroí uma lista com os valores de cada erro calculado.
50      verro.append(erro)
51
52      # Estabelece a avaliação do tempo decorrido durante a simulação.
53      T = watch.time()
54      T = T*0.001
55
56      # Calcula o valor a ser atrubuído durante a realização de uma curva.
57      curva = (PROPORTIONAL_GAIN * erro)
58
59      # Computa os valores do tempo, do erro e da curva em uma linha da tabela de dados.
60      data.log(T, erro, curva)
61
62      # Atribuí o valor da curva em conjunto com a velocidade desenvolvida pelos motores direito e esquerdo
63      mdireito.run(int(DRIVE_SPEED-curva))
64      mesquerdo.run(int(DRIVE_SPEED+curva))
65
66      # Espera 10 milissegundos para a realização da próxima leitura.
67      wait(10)
```

Fonte: Adaptado de The Cooding Fun (2020).

3.1.12 Controlador PI:

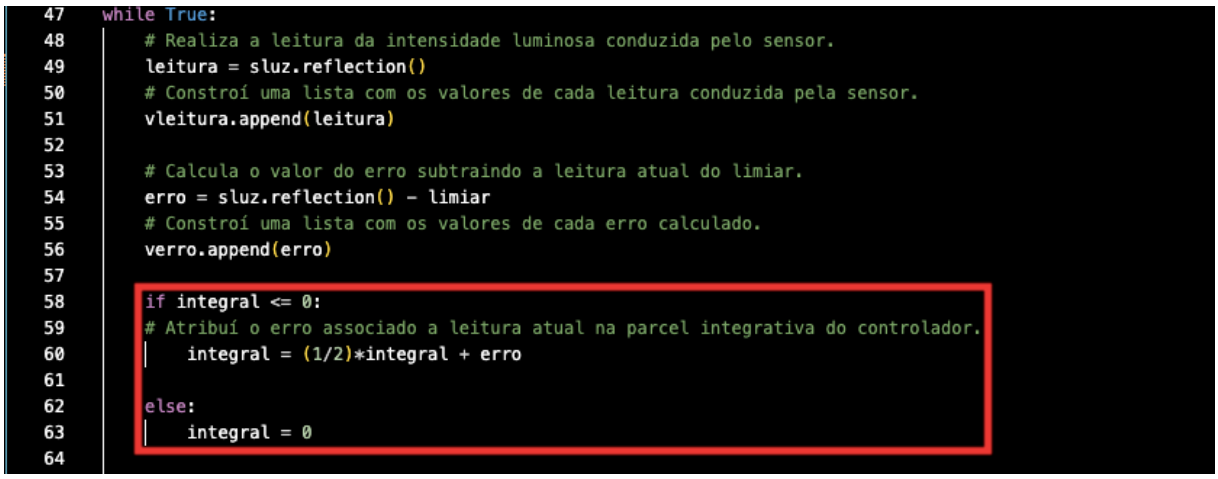
Para melhorar ainda mais o comportamento do robô ao longo do percurso é preciso mudar significativamente a estratégia. Na verdade, o controlador P apresenta um desempenho interessante até um certo limite. Posteriormente, a função responsável por mapear seu comportamento é insuficiente, caso o objetivo seja aumentar a precisão e a velocidade.

Uma alternativa válida é incorporar o caráter Integrativo ao controlador com a adição de um novo termo a função. Com a integral presente, é possível contabilizar um valor de erro a todo o momento em que o sensor fizer uma nova leitura. Assim, o erro se torna um elemento acumulativo no sistema.

No trecho de código apresentado na figura abaixo, a primeira seção destaca a composição do termo Integrativo no sistema. A expressão associada a ele permite armazenar o valor de erro presente na leitura atual e compará-lo com o erro presente na leitura anterior.

Logo, é uma expressão criada para viabilizar uma facilidade dentro da programação do robô. O código completo está presente na seção Apêndice (G).

Figura 85 – Lógica empregada na implementação do controlador PI.



Fonte: Adaptado de The Cooding Fun (2020).

Ou seja, essa contabilização dos erros pontuais é transformada em uma contabilização geral associada ao processo de leitura do sensor, tornando-se uma ótima alternativa para conduzir alguma readequação.

Entretanto, existe um grande obstáculo no processo de contabilização, caso não exista uma atenuação do erro característico. Como o termo Integrativo tem uma tendência inerente ao infinito, buscando evitar essa situação, ele foi multiplicado por um valor inferior a 1. Paralelamente, esse termo também está sujeito a uma mudança de sinal, ou seja, a tendência ao infinito pode ocorrer com valores positivos, ou com valores negativos.

Com o objetivo de evitar essa mudança brusca de sinal e manter o comportamento natural do controlador, toda vez que o termo assume um valor positivo devido a contabilização de erro, ele retorna para 0. Essa pequena adição no código geralmente é chamada de “damping” (Amortecimento em inglês).

Sendo assim, o controlador PI não foi construído em sua estrutura natural. Esses elementos implementados fazem com que seu comportamento se afaste levemente do ideal. Logo, é possível dizer que o controlador elaborado é análogo a um controlador PI.

Em termos da função responsável por direcionar a curva do robô construída anteriormente, ela ganha a seguinte forma:

$$Curva = (ganho\ proporcional) \cdot (erro) + (ganho\ integral) \cdot (integral) \quad (42)$$

Figura 86 – Expressão responsável por adequar a influência proporcional e integrativa do controlador.

```
65    # Estabelece a avaliação do tempo decorrido durante a simulação.
66    T = watch.time()
67    T = T*0.001
68
69    # Calcula o valor a ser atrubuído durante a realização de uma curva.
70    curva = (PROPORTIONAL_GAIN * erro) + (INTEGRAL_GAIN * integral)
71
72    # Computa os valores do tempo, do erro, da integral e da curva em uma linha da tabela de dados.
73    data.log(T, erro, integral, curva)
74
75    # Atribuí o valor da curva em conjunto com a velocidade desenvolvida pelos motores direito e esquerdo.
76    mdireito.run(int(DRIVE_SPEED-curva))
77    mesquerdo.run(int(DRIVE_SPEED+curva))
```

Fonte: Adaptado de The Cooding Fun (2020).

Vale ressaltar que nesse caso o termo integrador depende do tempo, pois cada umas das leituras feitas pelo sensor estão condicionadas a um instante de tempo distinto. Devido a essa característica das leituras, o integrador também deve realizar sua composição de acordo com a passagem de tempo estabelecida entre uma amostra e outra.

Porém, ao invés de deixar o termo associado ao tempo evidente na equação, é mais interessante deixá-lo implícito. Esse processo exige a criação de uma nova constante expressa da seguinte maneira:

$$K_i = Ki_0 \cdot dT \quad (43)$$

Essa pequena mudança facilita a composição da equação, deixando-a mais simples, como visto no código acima:

$$Curva = (K_p \cdot erro) + (K_i \cdot\ integral) \cdot dT \quad (44)$$

$$Curva = (K_p \cdot erro) + (K_i \cdot\ integral) \quad (45)$$

Como o termo da integral também apresenta um ganho, mas diferente daquele associado ao erro, uma nova constante é criada para designá-lo. Essa nova estrutura da função permite que o controlador seja capaz de avaliar o erro presente em leituras atuais (Controle Proporcional), como também em leituras passadas (Controlador Integral).

3.1.13 Controlador PD:

O controlador PD possui um comportamento análogo ao do controlador PI, ou seja, apresenta características mais robustas em relação aos controladores anteriores, mas utiliza uma estratégia oposta a empregada pelo controlador PI.

Ele incorpora o caráter Derivativo ao controlador P, com a adição de um novo termo a função. Seu grande diferencial é a capacidade de contabilizar o erro em leituras futuras. Essa contabilização é feita levando em consideração o valor do erro atual e o erro entre os dois últimos valores computados pelo sensor.

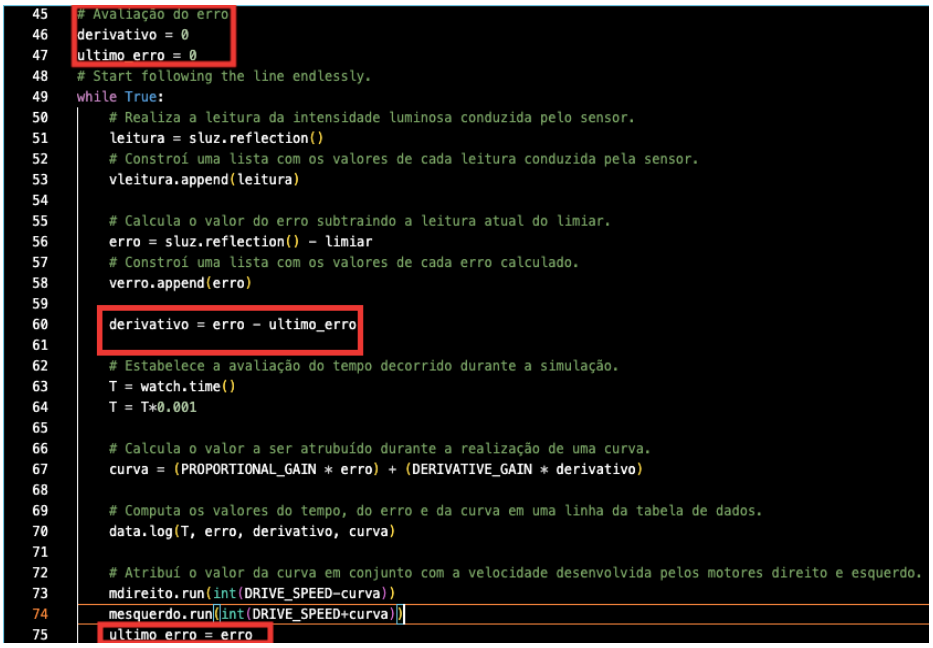
Desse modo, o objetivo é tentar mapear a ocorrência de erros a ponto de poder prevê-los idealmente. O erro associado ao termo Derivativo é dado pela seguinte estrutura:

$$Erro\ da\ Amostra\ Futura = Erro\ da\ Amostra\ Atual - \ Erro\ Derivativo \quad (46)$$

$$Erro\ Derivativo = Erro\ da\ Amostra\ Atual - Erro\ da\ Amostra\ Futura \quad (47)$$

No trecho de código apresentado na figura abaixo a primeira seção destaca a composição do termo Derivativo no sistema.

Figura 87 – Lógica empregada na implementação do controlador PD.



```
45  # Avaliação do erro
46  derivativo = 0
47  ultimo_erro = 0
48  # Start following the line endlessly.
49  while True:
50      # Realiza a leitura da intensidade luminosa conduzida pelo sensor.
51      leitura = sluz.reflection()
52      # Constroí uma lista com os valores de cada leitura conduzida pela sensor.
53      vleitura.append(leitura)
54
55      # Calcula o valor do erro subtraindo a leitura atual do limiar.
56      erro = sluz.reflection() - limiar
57      # Constroí uma lista com os valores de cada erro calculado.
58      verro.append(erro)
59
60      derivativo = erro - ultimo_erro
61
62      # Estabelece a avaliação do tempo decorrido durante a simulação.
63      T = watch.time()
64      T = T*0.001
65
66      # Calcula o valor a ser atrubuído durante a realização de uma curva.
67      curva = (PROPORTIONAL_GAIN * erro) + (DERIVATIVE_GAIN * derivativo)
68
69      # Computa os valores do tempo, do erro e da curva em uma linha da tabela de dados.
70      data.log(T, erro, derivativo, curva)
71
72      # Atribuí o valor da curva em conjunto com a velocidade desenvolvida pelos motores direito e esquerdo.
73      mdireito.run(int(DRIVE_SPEED-curva))
74      mesquerdo.run(int(DRIVE_SPEED+curva))
75      ultimo_erro = erro
```

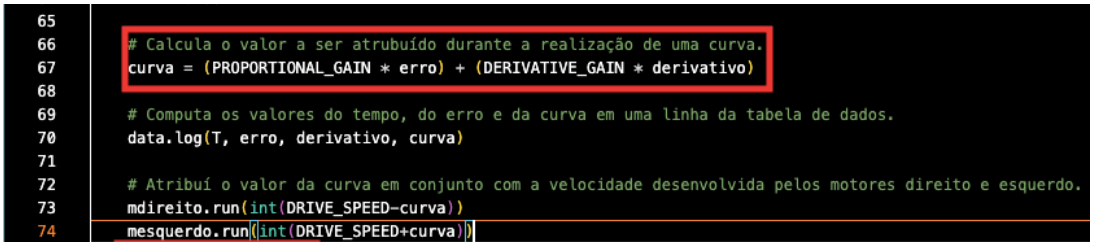
Fonte: Adaptado de The Cooding Fun (2020).

A expressão associada a ele permite prever o valor de erro presente na leitura futura a partir da comparação com o erro presente nas leituras anteriores. Logo, é uma expressão criada para viabilizar uma facilidade dentro da programação do robô.

Em termos da função responsável por direcionar a curva do robô construída anteriormente, ela ganha a seguinte forma:

$$Curva = (ganho\ proporcional) \cdot (erro) + (ganho\ derivativo) \cdot (derivada) \tag{48}$$

Figura 88 – Expressão responsável por adequar a influência proporcional e derivativa do controlador.



```
65
66      # Calcula o valor a ser atrubuído durante a realização de uma curva.
67      curva = (PROPORTIONAL_GAIN * erro) + (DERIVATIVE_GAIN * derivativo)
68
69      # Computa os valores do tempo, do erro e da curva em uma linha da tabela de dados.
70      data.log(T, erro, derivativo, curva)
71
72      # Atribuí o valor da curva em conjunto com a velocidade desenvolvida pelos motores direito e esquerdo.
73      mdireito.run(int(DRIVE_SPEED-curva))
74      mesquerdo.run(int(DRIVE_SPEED+curva))
```

Fonte: Adaptado de The Cooding Fun (2020).

Assim como nos casos anteriores, o termo derivativo além de estar submetido a um ganho, também é dependente da passagem de tempo. O código completo está na seção Apêndice (H).

Novamente, o termo associado ao tempo foi deixado de forma implícita, exigindo a criação de uma nova constante:

$$Kd = \frac{Kd_0}{dT} \tag{49}$$

Após a realização desse processo é possível encontrar a estrutura geral de um controlador PD:

$$Curva\ = (K_p \cdot erro) + (K_d \cdot derivativo) \tag{50}$$

Como o termo da derivada também apresenta um ganho, mas diferente daquele associado ao erro, uma nova constante é criada para designá-lo. Essa nova estrutura da função permite que o controlador seja capaz de avaliar o erro presente em leituras atuais (Controle Proporcional), como também em leituras futuras (Controlador Derivativo).

3.1.14 Controlador PID:

Após a construção dos controladores apresentados acima, foi conduzida a elaboração do controlador PID, isto é, um controlador extremamente robusto, sendo capaz de condensar as características dos controladores P, PI e PD em sua própria estrutura. Desse modo, sua a estrutura geral é um aglutinado das estruturas de cada um deles:

$$Curva\ = (K_p \cdot erro) + (K_i \cdot integral) + (K_d \cdot derivativo) \tag{51}$$

Portanto, essa função permite que o controlador seja capaz de avaliar o erro presente em leituras atuais (Controle Proporcional), em leituras passadas (Controle Integral), como também em leituras futuras (Controle Derivativo). O código completo está presente na seção Apêndice (I).

A imagem seguinte ilustra essa lógica implementada em código:

Figura 89 – Lógica empregada na implementação do controlador PID.

```
while True:
    # Realiza a leitura da intensidade luminosa conduzida pelo sensor.
    leitura = sluz.reflection()
    # Constrói uma lista com os valores de cada leitura conduzida pela sensor.
    vleitura.append(leitura)

    # Calcula o valor do erro subtraindo a leitura atual do limiar.
    erro = sluz.reflection() - limiar
    # Constrói uma lista com os valores de cada erro calculado.
    verro.append(erro)

    # Composição de erro avaliando o termo integral e o termo derivativo.
    if integral <= 0:
    # Atribuí o erro associado a leitura atual na parcel integrativa do controlador.
        integral = (1/2)*integral + erro

        if integral > 0:
            integral = 0

    else:
        integral = 0

    derivativo = erro - ultimo_erro

    # Estabelece a avaliação do tempo decorrido durante a simulação.
    T = watch.time()
    T = T*0.001

    # Cálculo de curva em função de cada um dos termos:
    curva = (PROPORTIONAL_GAIN * erro) + (INTEGRAL_GAIN * integral) + (DERIVATIVE_GAIN * derivativo)
    # Contabilização dos valores na tabela:
    data.log(T, erro, integral, derivativo, curva)

    # Atribuí o valor da curva em conjunto com a velocidade desenvolvida pelos motores direito e esquerdo.
    mdireito.run(int(DRIVE_SPEED+curva))
    mesquerdo.run(int(DRIVE_SPEED-curva))
    ultimo_erro = erro

    # Espera 10 milissegundos para a realização da próxima leitura.
    wait(10)
```

Fonte: Adaptado de The Cooding Fun (2020).

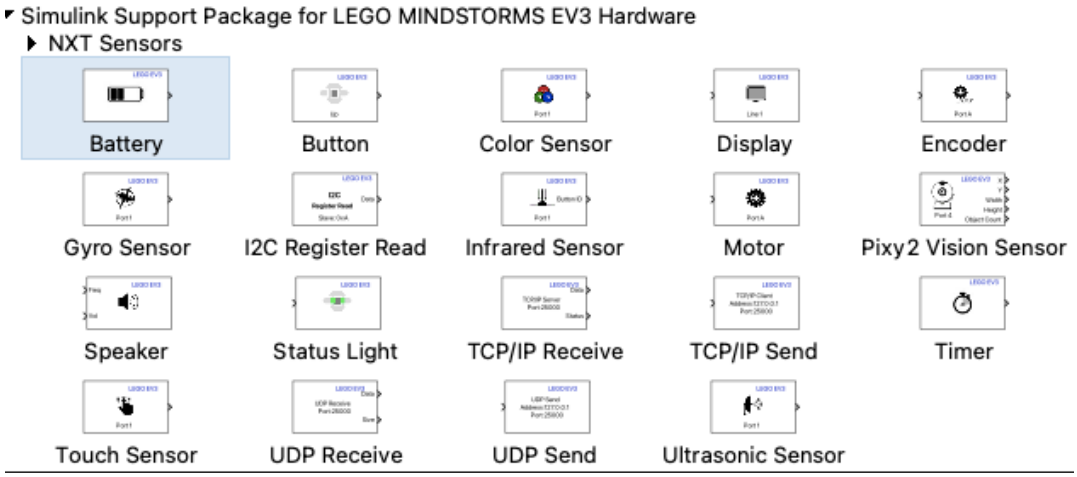
3.1.15 Simulação no ambiente virtual SIMULINK:

Em conjunto com a simulação realizada no protótipo, também foi desenvolvida uma simulação virtual utilizando o pacote de conteúdo “Mobile Robotics Training”. Esse pacote de conteúdo disponibiliza uma variedade de recursos para a construção de diagramas de bloco responsáveis por representar o funcionamento do sistema dinâmico em análise.

Ele também é compatível com o pacote de conteúdo “LEGO MINDSTORMS EV3 Hardware”. Essa compatibilidade viabilizou a utilização do software para realizar algumas simulações mais simples, visando verificar se o comportamento dos blocos dentro dos diagramas criados estava de acordo com o comportamento real do robô seguidor de linha.

A Figura 90 e a Figura 91 mostram uma visão geral dos blocos presentes em cada pacote.

Figura 90 – Blocos disponíveis no pacote "LEGO MINDSTORMS EV3 Hardware".



Fonte: Autor.

Figura 91 – Blocos disponíveis no pacote "Mobile Robotics Training".

Fonte: Autor.

Esse procedimento foi necessário, visto que os códigos criados para os controladores P, PI, PD e PID necessitam de uma calibração dos ganhos proporcional, integral e derivativo, contudo o Visual Studio Code não possui os recursos exigidos para determiná-los com base na função de transferência do sistema.

Sendo assim, essa calibração foi feita se valendo da construção de diagramas de blocos correspondente ao robô seguidor de linha, assim como aos controladores empregados.

3.1.16 Configuração do trajeto:

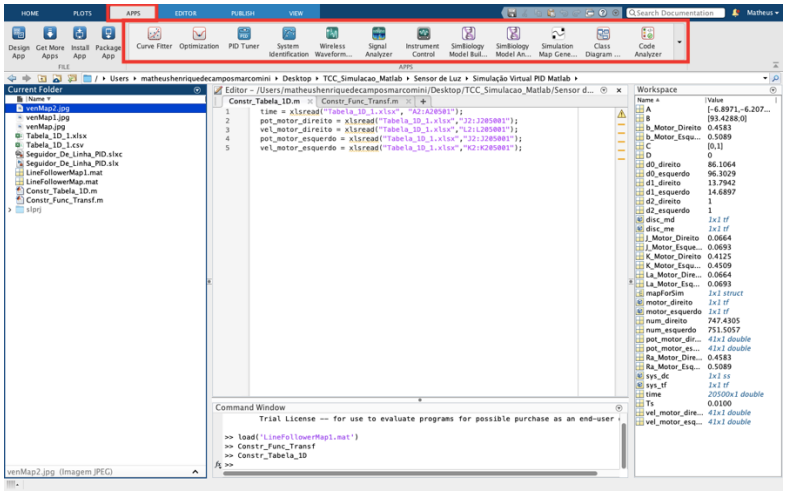
Outro detalhe bastante importante é o trajeto a ser percorrido pelo robô. Para criar uma paridade entre os testes reais e os testes virtuais, um trajeto foi definido, e transferido por meio de digitalização.

No ambiente virtual do Matlab, a digitalização utiliza como base uma lógica de análise de imagem. Essa imagem pode ser tirada utilizando uma câmera fotográfica, ou uma câmera presente em um dispositivo móvel.

Esse processo de digitalização da imagem é um outro recurso disponibilizado pelo pacote de conteúdo “Mobile Robotics Training” instalado no Matlab. Diferentemente das outras funcionalidades, essa é acessada por meio da aba “APPS” a partir do ícone “Simulation Map Generator”.

A figura a seguir ilustra a tela inicial do Matlab com destaque para a aba “APPS” onde o aplicativo está localizado.

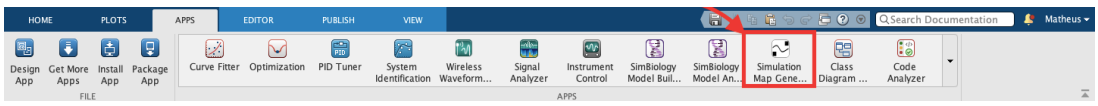
Figura 92 – Tela inicial do software Matlab.



Fonte: Autor.

Já a Figura 93 mostra a barra de seleção de aplicativos com destaque para o ícone do “Simulation Map Generator”.

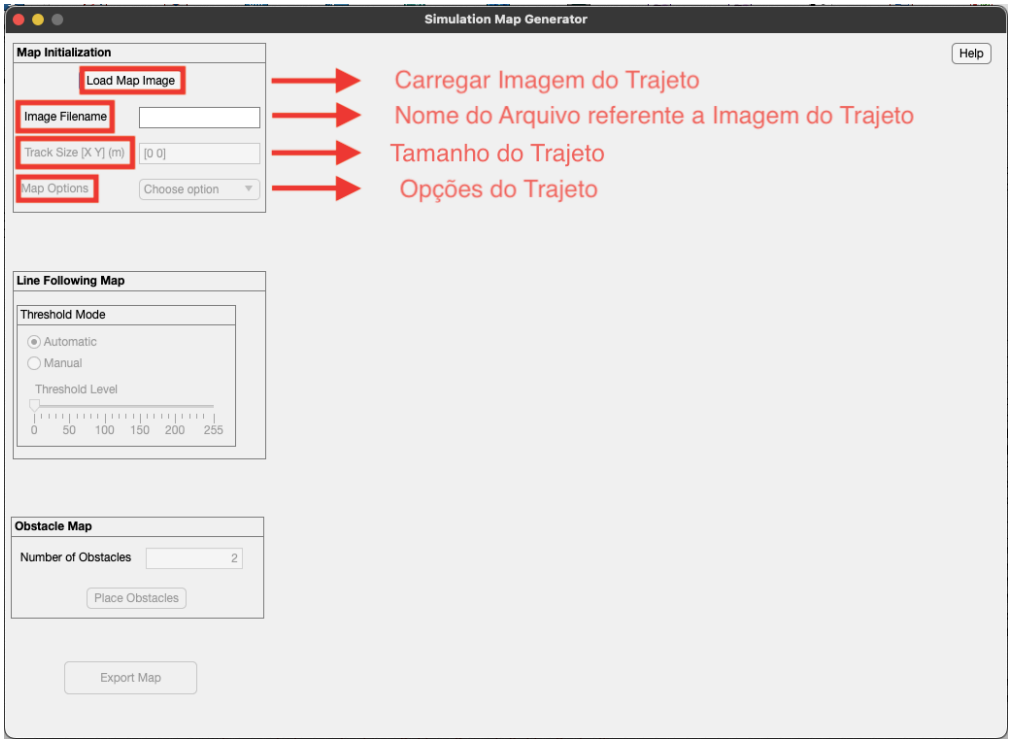
Figura 93 - Ícone do aplicativo “Simulation Map Generator”.



Fonte: Autor.

Com a aba aberta, a imagem foi carregada selecionando o botão “Load Map Image”. Vale ressaltar que o aplicativo permite o uso de arquivos somente no formato (.jpg), ou seja, caso não esteja nesse padrão de extensão é necessário convertê-la. A Figura (94) demonstra esse procedimento:

Figura 94 – Tela de configuração do trajeto.



Fonte: Autor.

Em seguida, o tamanho real do trajeto é solicitado. Sendo assim, as respectivas dimensões do eixo x (horizontal) e do eixo y (vertical) foram obtidas com o uso de uma fita métrica.

A dimensão do eixo x corresponde a 3 metros, enquanto a dimensão do eixo y corresponde a 1,5 metros. Foi observado que imagens abaixo dessa medida acarretam uma escala não proporcional entre o tamanho da linha e o tamanho do robô na simulação.

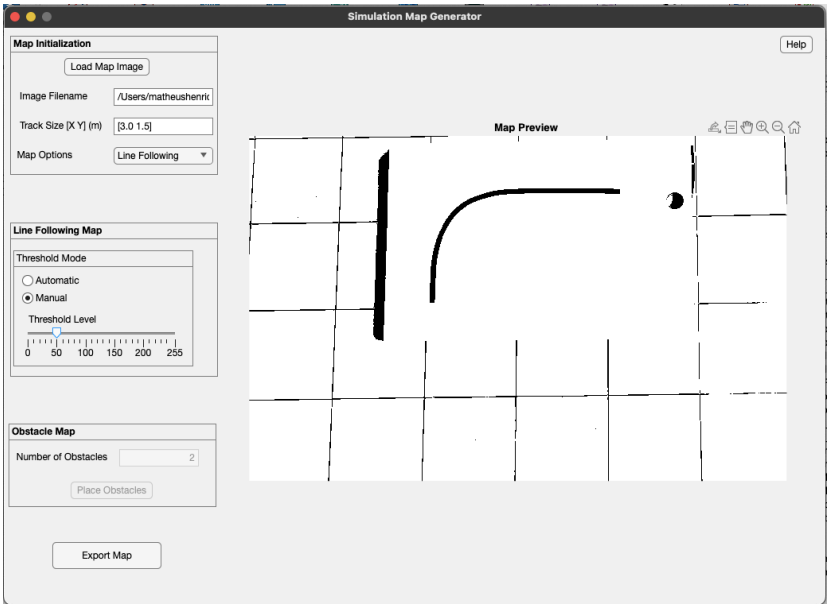
O trajeto a ser percorrido foi projetado com base em uma situação em que o robô fosse empregado no chão de fábrica, deslocando-se dentro de um circuito, envolvendo trecho retilíneo e curvilíneo.

O último item solicitado nas configurações, é o objetivo principal do trajeto, ou seja, o seguimento de linha, pois também existe a opção de configurar o trajeto para desvio de obstáculos, ou até mesmo ambas as situações.

Na seção “Line Following Map” o ícone “Threshold” foi modificado para a opção “Manual”, uma vez que as linhas presentes na imagem precisam ficar em destaque na coloração preta, e os demais elementos do ambiente precisam ficar em destaque na coloração branca.

Ao aumentar o valor da barra de ajuste a imagem ficará cada vez mais escura, e ao diminuir o valor da barra de ajuste a imagem ficará cada vez mais clara. Logo, a barra de ajuste foi reduzida próximo ao valor 30 na escala disponível. A Figura 95 destaca o resultado do procedimento de configuração:

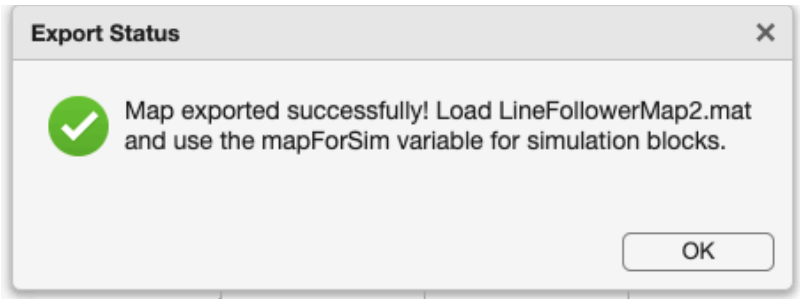
Figura 95 – Configuração das características do trajeto.



Fonte: Autor.

Com todas as alterações feitas, o trajeto foi exportado selecionando a opção “Export Map”, e o resultado é a criação de um objeto do tipo (.map) na pasta de origem, sendo carregado no ambiente virtual em seguida. A mensagem abaixo é exibida para demonstrar que o procedimento foi concluído com sucesso:

Figura 96 - Mensagem de conclusão.



Fonte: Autor.

3.1.17 Configuração das características do robô:

Posteriormente, um diagrama de blocos foi criado diretamente no SIMULINK para representar o sistema do robô seguidor de linha. Dessa forma, foi adicionado um conjunto de blocos, para representar as características físicas do robô, além de seu comportamento no ambiente virtual.

O primeiro bloco implementado foi o “Line Sensor Simulation”. Ele é responsável por representar o Sensor de Luz utilizado na montagem do robô. Para o bloco apresentar um comportamento análogo ao do sensor empregado, os parâmetros “Map Variable”, “Environment Value”, e “Line Value” foram modificados conforme ilustrado pela figura a seguir.

Figura 97 – Parâmetros do bloco "Line Sensor".

Fonte: Autor.

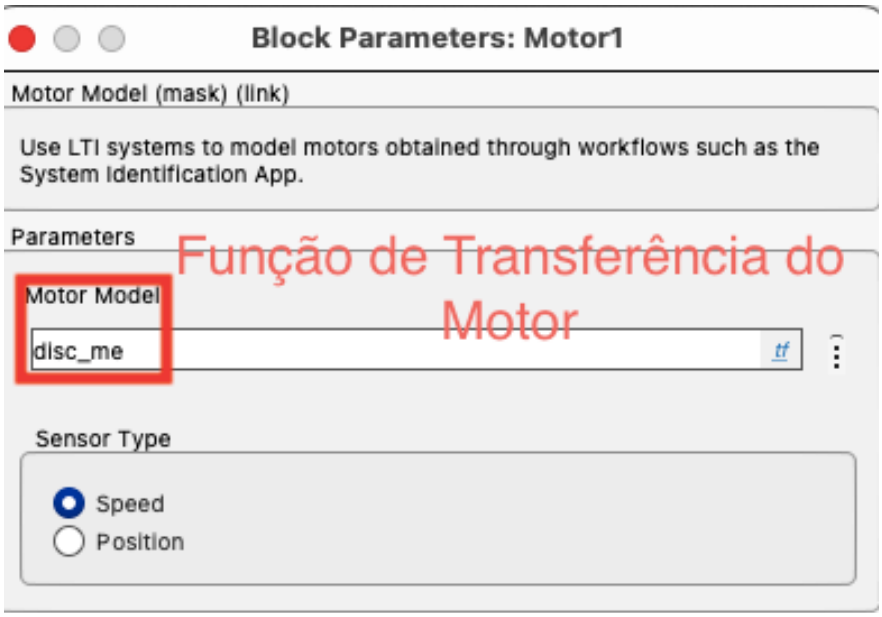
Os valores atribuídos as características do sensor “Enviroment Value” e “Line Value” foram obtidos de maneira empírica, assim como destacado anteriormente. Os valores foram preenchidos com a respectiva porcentagem de cada coloração, isto é, 5% para fita isolante preta e 50% para a base branca.

A ideia central é que o robô faça uma leitura e compute um valor. Por exemplo, quando aplicado o controlador ON/OFF, caso o valor esteja acima de 27,5% ele fará uma curva para a direita. Caso, o valor esteja abaixo de 27,5% fará uma curva para a esquerda.

O bloco “Motor” recebe uma construção relativamente diferente. A sua entrada é um valor associado a velocidade do motor, enquanto sua saída é um valor associado a uma porcentagem da potência do motor. Dessa maneira, o bloco “1-D Dimension Lookup Table” equivalente a uma tabela fica encarregado de criar essa relação para transformar valores de velocidade em valores de porcentagem de potência.

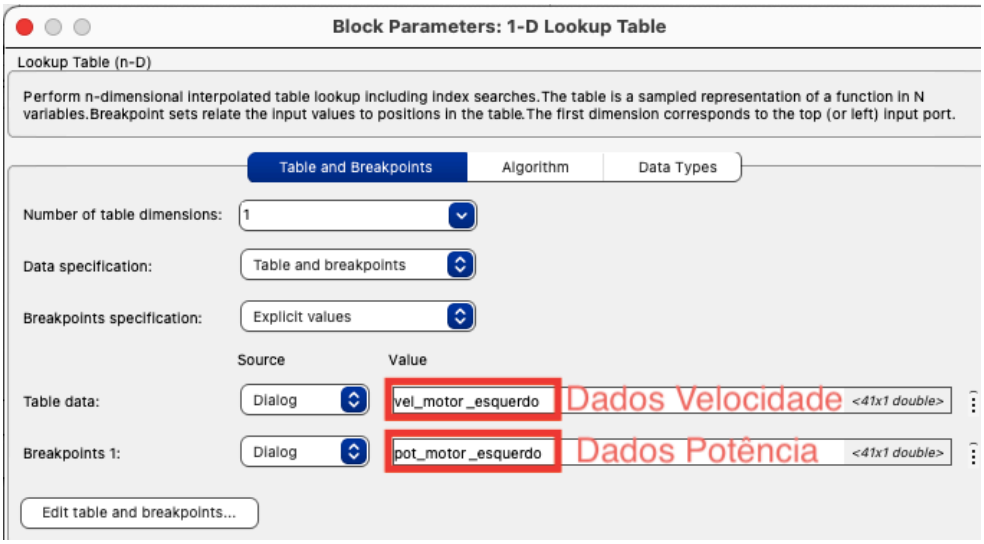
Como já se tinha esses dados e informações devido ao modelamento matemático dos motores, a criação dessa tabela foi feita a partir deles. As figuras seguintes exibem as configurações disponíveis no bloco “Motor” e no bloco “1-D Dimension Lookup Table”.

Figura 98 – Configuração do bloco "Motor".



Fonte: Autor.

Figura 99 – Configuração do bloco "Motor".



Fonte: Autor.

As configurações, propriamente ditas, do bloco “Motor” solicitam a seleção de um modelo matemático para compor a referência de comportamento. Essa referência foi criada com base na função de transferência selecionada na seção 3.1.8.

Foi desenvolvido um código para deixá-la disponível para seleção. O código completo utilizado para a construção da função de transferência do motor e da "1-D Dimension Lookup Table" está presente na seção Apêndice (J) e Apêndice (K), respectivamente.

A imagem seguinte mostra um trecho do código responsável por disponibilizar a função de transferência.

Figura 100 – Código construído para disponibilizar a função de transferência no SIMULINK.

```
% Construção da Função de Transferência Utilizando os Valores Obtidos Por
% Meio da Simulação dos Valores de DataLog Computados Em Relação Ao Motor
% Direito e Em Relação Ao Motor Esquerdo:

% Atribuição Dos Valores Associados A Resistência Elétrica de Armadura
% (Ra), A Indutância Elétrica de Armadura (La), A Constante de Torque ou A
% Constante de Tensão Elétrica de Retorno (K), O Momento de Inércia (J) e O
% Coeficiente de Amortecimento:

% Motor Direito:
Ra_Motor_Direito = 0.45827;
La_Motor_Direito = 0.066444;
K_Motor_Direito = 0.41247;
J_Motor_Direito = 0.066444;
b_Motor_Direito = 0.45827;

% Constantes Função de Transferência do Motor Direito:
num_direito = 8*(K_Motor_Direito/(J_Motor_Direito*La_Motor_Direito));
d0_direito = ((b_Motor_Direito*Ra_Motor_Direito)+(K_Motor_Direito* ...
    K_Motor_Direito))/(J_Motor_Direito*La_Motor_Direito);
d1_direito = ((b_Motor_Direito*La_Motor_Direito)+ ...
    (J_Motor_Direito*Ra_Motor_Direito))/(J_Motor_Direito*La_Motor_Direito);
d2_direito = 1;

% Motor Esquerdo:
Ra_Motor_Esquerdo = 0.50886;
La_Motor_Esquerdo = 0.069281;
K_Motor_Esquerdo = 0.45089;
J_Motor_Esquerdo = 0.069281;
b_Motor_Esquerdo = 0.50886;

```

Fonte: Autor.

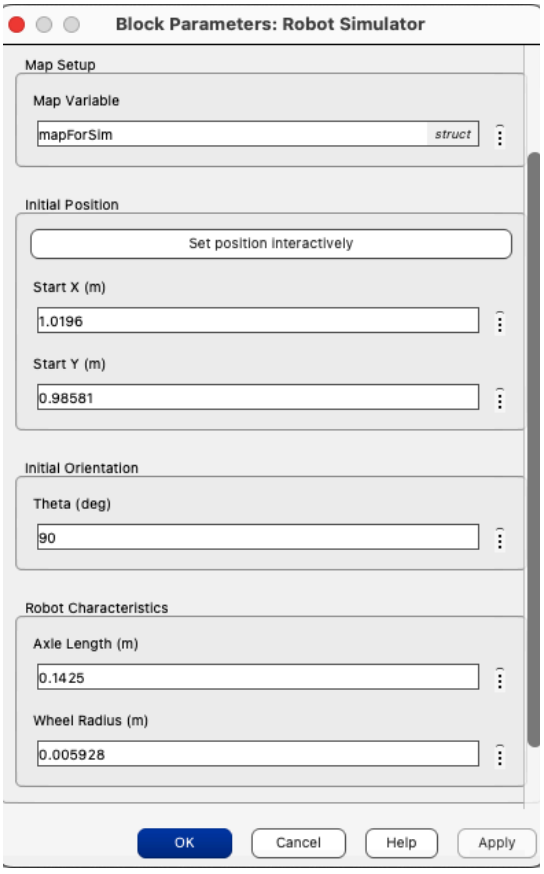
Cada motor recebeu uma função de transferência distinta, entretanto essa distinção das funções de transferência impactava diretamente no movimento do robô. Com um comportamento específico atribuído a cada motor, em alguns casos pode ser prejudicial para a realização de uma atividade específica. Logo, a função selecionada como referência foi:

$$G(s) = \frac{751.5}{s^2 + 14.69s + 96.3} \tag{52}$$

Essa função corresponde ao motor esquerdo sem rodas funcionando com uma potência de 100%. O critério utilizado para selecioná-la foi o fato de o motor esquerdo aparentar ser mais rápido em relação ao motor direito durante os testes de funcionamento. Desse modo, essa função de transferência se adequou melhor a simulação.

A próxima etapa foi adicionar o bloco “Robot Simulator” e configurar os parâmetros “Map Variable”, “Initial Position”, “Axle Length” e “Wheel Radius”.

Figura 101 - Parâmetros do bloco "Robot Simulator".

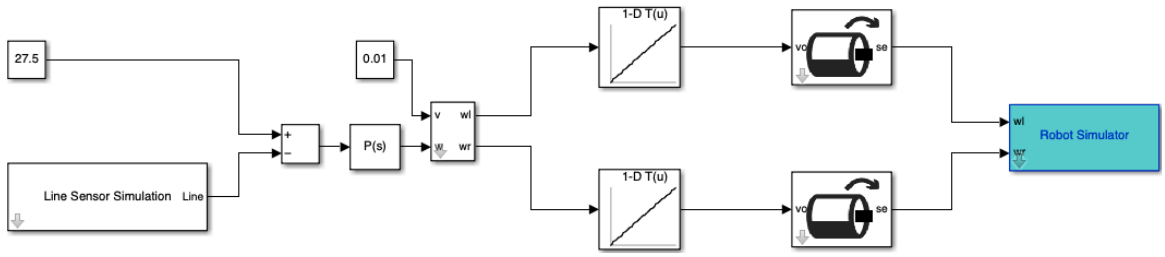


Fonte: Autor.

Por último, também houve a necessidade de inserir o algoritmo no caso do controlador ON – OFF e do controlador de 3 Níveis. O algoritmo é inserido por meio de uma estrutura chamada de “Stateflow Chart”.

Para os controladores P, PI, PD e PID ao invés de um algoritmo foi inserido um bloco. Nesse bloco, com o auxílio da ferramenta “PID tuner” tornou-se relativamente mais fácil avaliar quais valores de ganho deveriam ser atribuídos a cada termo dos controladores.

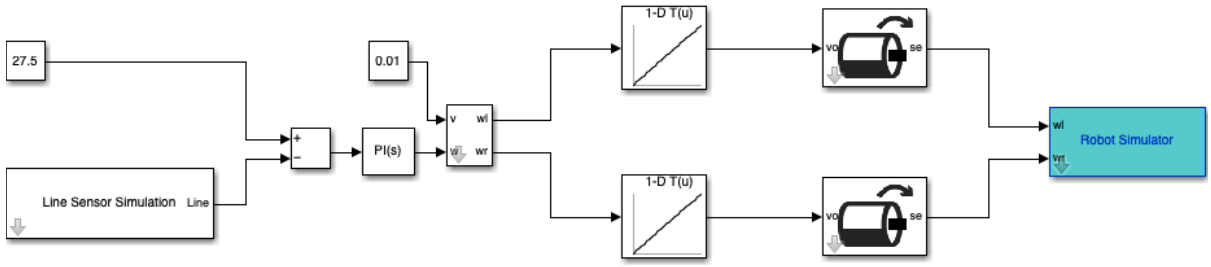
Figura 102 - Diagrama de blocos do sistema com controlador P.



Fonte: Autor.

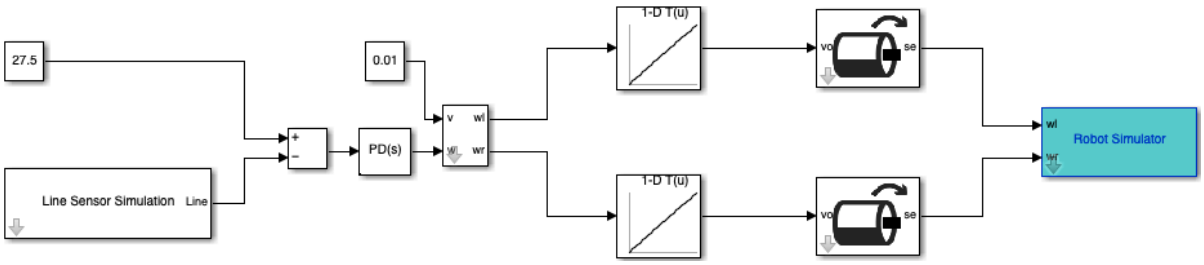
Os diagramas presentes na Figura 102, 103, 104 são uma variação do diagrama presente na figura 101. A única diferença entre cada um deles é o bloco do controlador empregado.

Figura 103 – Diagrama de blocos do sistema e do controlador PI.



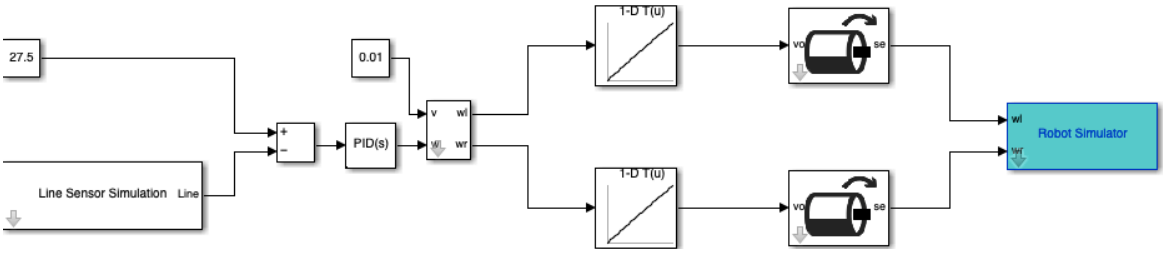
Fonte: Autor.

Figura 104 – Diagrama de blocos do controlador PD.



Fonte: Autor.

Figura 105 – Diagrama de blocos do controlador PID.



Fonte: Autor.

3.1.18 Calibração dos ganhos:

Nessas situações, ainda é fundamental estabelecer um elemento capaz de fazer o sensor considerar a borda da linha preta como referência, isto é, o trajeto a ser percorrido. Contudo, cada controlador atribui características particulares ao deslocamento do robô.

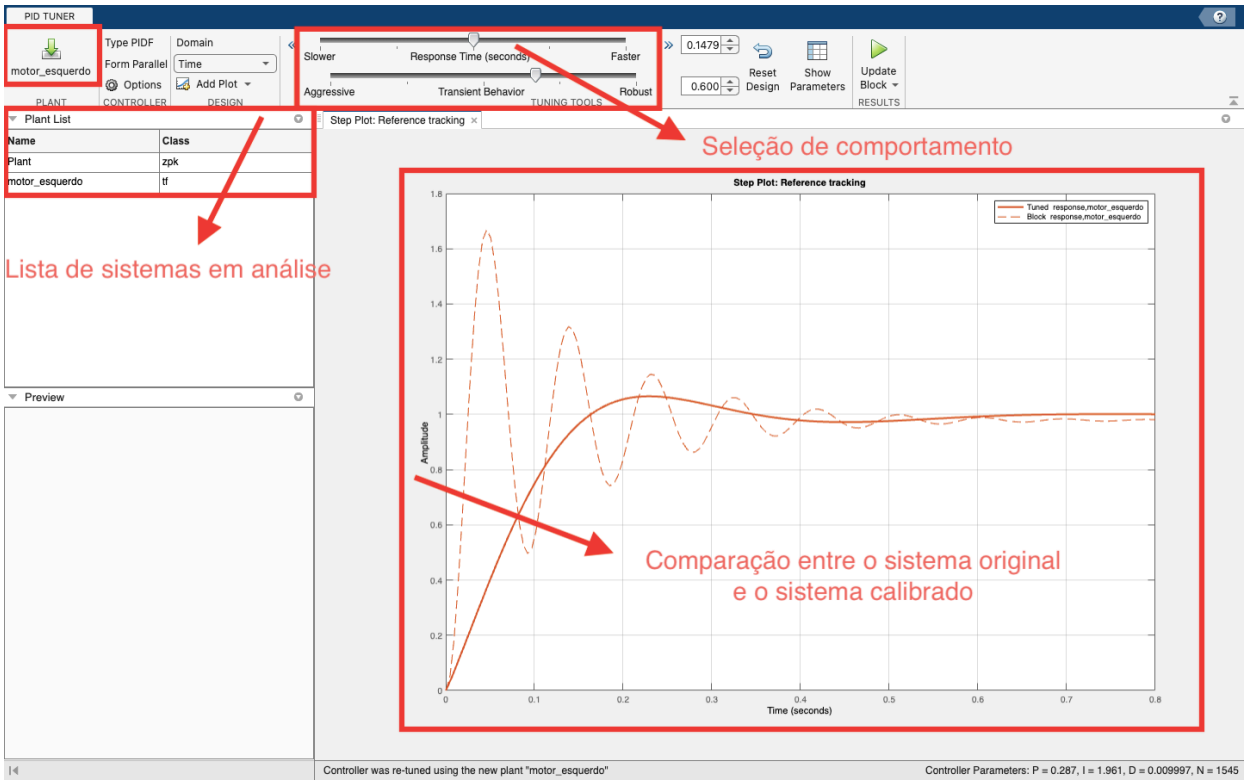
Os controladores ON/OFF e de 3 Níveis possuem um elemento lógico capaz de garantir o funcionamento adequado do robô seguidor de linha, contudo os controladores P, PI, PD e PID exigem o estabelecimento dos respectivos ganhos associado a cada comportamento.

Desse modo, foi preciso criar uma lógica para a calibração desses valores, buscando a aproximação de um ponto ótimo. Existem várias formas de conduzir esse procedimento de calibração utilizando metodologias mais simples como a tentativa e erro, ou até mesmo metodologias mais complexas como Ziegler-Nichols.

Como o Matlab possui a funcionalidade chamada de “PID tuner” é possível fazer essa calibração de um modo interativo, observando as mudanças no comportamento da função conforme alguns dos valores de seus parâmetros são modificados.

A grande facilidade em utilizar essa ferramenta presente no SIMULINK é o fato dela garantir a estabilidade do sistema em análise, além de destacar se o comportamento do controlador permitirá um caráter mais rápido, ou mais lento da reposta, por exemplo. A figura seguinte trás uma visão geral da interface da ferramenta.

Figura 106 – Tela de configuração do PID tuner.



Fonte: Autor.

Assim, é possível verificar a faixa de valores disponíveis para cada ganho, e consequentemente o comportamento ideal na janela de simulação do robot simulator. Posteriormente, esses mesmos valores puderam constituir os melhores pontos de partida para a simulação no protótipo.

4 RESULTADOS E DISCUSSÃO:

No decorrer desta seção, será elaborada uma visão completa e equilibrada das descobertas, destacando os pontos fortes e os pontos fracos, além das limitações de cada controlador, buscando estabelecer uma análise de cunho tanto qualitativo, quanto quantitativo.

Para complementar esse conteúdo, foram empregadas técnicas estatísticas com o objetivo de consolidar resultados mais factíveis matematicamente. Eles serão fundamentais para a conclusão do trabalho, permitindo que se chegue a avaliações mais sólidas e respaldadas pela análise rigorosa dos dados.

Além disso, servirão como base para recomendações e sugestões relevantes para futuros aprimoramentos e pesquisas na área de automação, controle e robótica seguidora de linha.

A simulação virtual foi utilizada como referência do comportamento que o robô deveria assumir na simulação real, ou seja, o protótipo construído. Desse modo, no caso dos controladores ON/OFF e de 3 Níveis, o desempenho foi medido em termos da oscilação executada pelo robô conforme ele avançava sobre a linha, realizando novas leituras.

A ferramenta “PID tuner” presente no bloco apresentado na seção 3.1.18 foi uma saída encontrada para estabelecer o melhor valor, ou a melhor faixa de valores a serem implementados no protótipo em funcionamento em relação aos ganhos associados aos controladores P, PI, PD e PID.

Entretanto, foi possível identificar algumas diferenças entre os valores determinados a partir da simulação virtual e a partir da simulação real.

4.1 Resultados da simulação virtual:

Após a realização dos procedimentos apresentados na seção Metodologia, um conjunto de resultados foi obtido. No caso da simulação virtual, os resultados mais importantes estão relacionados com a faixa dos valores de ganho.

4.1.1 Controlador ON/OFF e de 3 Níveis:

Os controladores ON/OFF e de 3 Níveis não puderam ser avaliados da mesma forma que o controlador PID e seus derivados. Como a lógica deles depende de um “Stateflow Chart”, não existe a necessidade de calibrar o ganho.

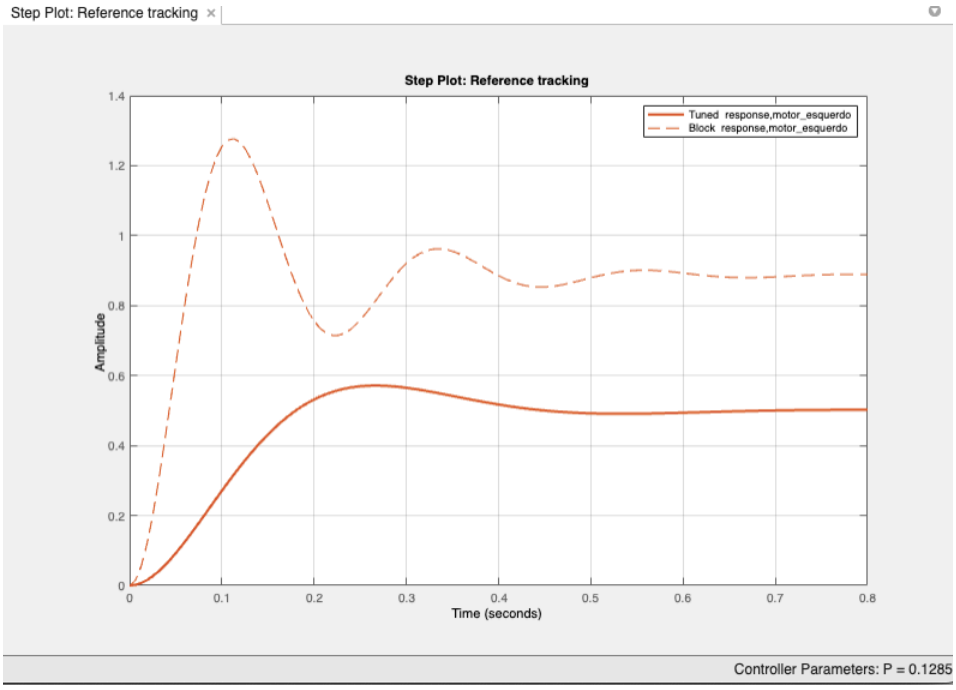
Outro detalhe é que a lógica presente nesses diagramas foi transformada diretamente em linhas de código, portanto o comportamento virtual e comportamento real são praticamente idênticos.

Para os demais controladores houve o uso da ferramenta “PID tuner”. Sendo assim, a calibração dos ganhos foi otimizada, uma vez que com a função de transferência em mãos (Equação (52)), o software conseguia fornecer um intervalo de valores específicos, para o qual o sistema se mantinha estável.

4.1.2 Controlador Proporcional (P):

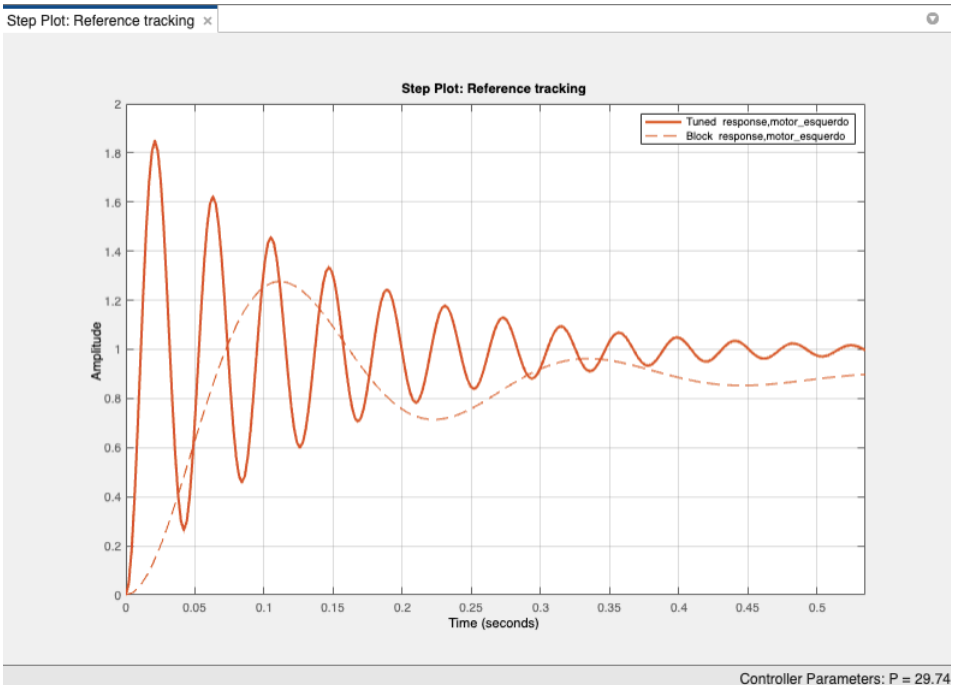
No caso do Controlador Proporcional (P) a faixa de valores determinada pelo uso do “PID tuner” foi de 0.1254 até 29.7422. Os gráficos a seguir ilustram o comportamento da função de transferência do sistema para essas situações:

Gráfico 1 – Resposta do sistema associado a um controlador P (0.1285) com resposta lenta.



Fonte: Autor.

Gráfico 2 – Resposta do sistema associado a um controlador P (29.74) com resposta rápida.



Fonte: Autor.

Ao avaliar os dois gráficos acima, é evidente que o comportamento entre eles é bastante distinto. O Gráfico 1 possui um caráter "mais comportado", portanto é mais simples de identificar a passagem do regime transitório para o regime permanente. Enquanto, o Gráfico 2 possui um caráter "menos comportado", portanto é mais difícil identificar a passagem do regime transitório para o regime permanente.

Mais precisamente, essa identificação é uma decorrência do baixo número de oscilações durante o regime transitório do comportamento mais lento, e do elevado número de oscilações no regime transitório durante o comportamento mais rápido.

Outras diferenças mais específicas podem ser visualizadas nas tabelas abaixo, que abordam informações como o tempo de subida (Rising Time), o tempo de acomodação (Settling Time), o sobressinal (Overshoot), o tempo de pico (Peak), entre outros.

Tabela 16 – Parâmetros controlador P com resposta lenta.

Performance and Robustness

	Tuned
Rise time	0.122 seconds
Settling time	0.416 seconds
Overshoot	14.1 %
Peak	0.571
Gain margin	Inf dB @ Inf rad/s
Phase margin	167 deg @ 1.49 rad/s
Closed-loop stability	Stable

Fonte: Autor

Tabela 17 – Parâmetros controlador P com resposta rápida.

Performance and Robustness

	Tuned
Rise time	0.00722 seconds
Settling time	0.527 seconds
Overshoot	85.7 %
Peak	1.85
Gain margin	Inf dB @ Inf rad/s
Phase margin	5.64 deg @ 149 rad/s
Closed-loop stability	Stable

Fonte: Autor.

Na Tabela 16 o comportamento mais lento promove a ocorrência de um tempo de subida e de um tempo de acomodação demorados quando comparados com os mesmos parâmetros na Tabela 17, ou seja, associados ao comportamento mais rápido.

Logo, fica evidente que o aumento no ganho proporcional produziu um impacto significativo na diminuição desses valores. Em contrapartida, o máximo sobressinal foi elevado, assim como o tempo de pico. Ou seja, o controlador mais rápido apresenta valores mais altos se comparado ao controlador mais lento.

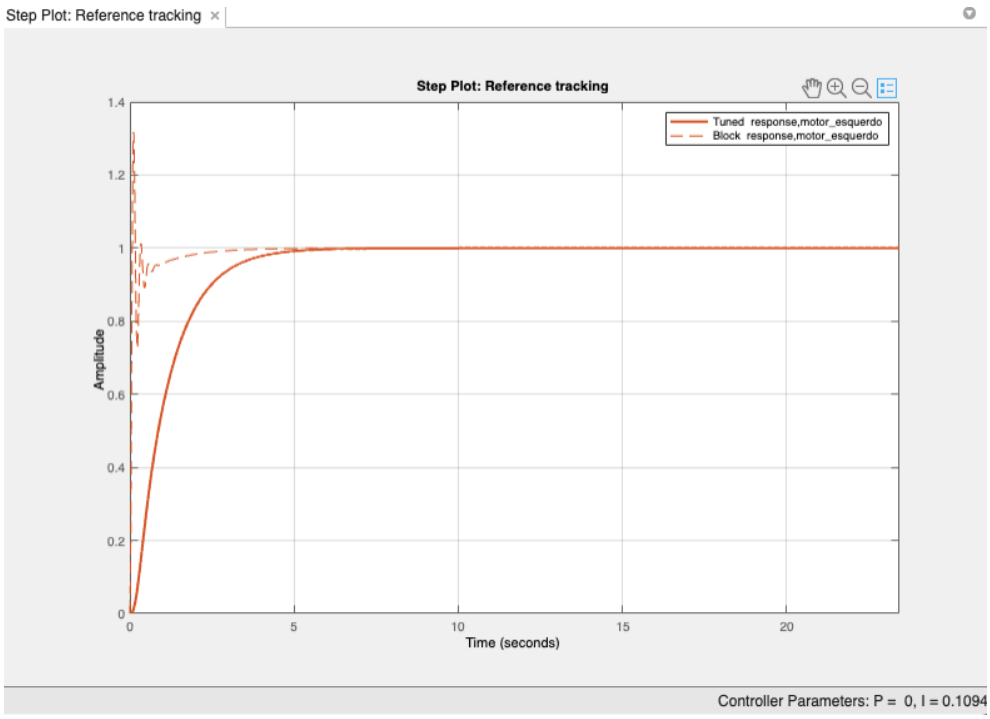
4.1.3 Controlador Proporcional-Integrativo (PI):

No caso do Controlador Proporcional-Integral (PI) a faixa de valores determinada pelo uso do “PID tuner” tem uma interdependência entre os ganhos. Dessa forma, existe uma diferença conforme o ganho proporcional é ajustado, assim como o ganho integral.

Foi possível perceber que ao manter um comportamento mais agressivo, o novo controlador permite que o Ganho P esteja compreendido dentro de um intervalo de valores entre 0 e 9.70265, enquanto o Ganho I esteja compreendido dentro de um intervalo de valores entre 0.10944 e 142.3118.

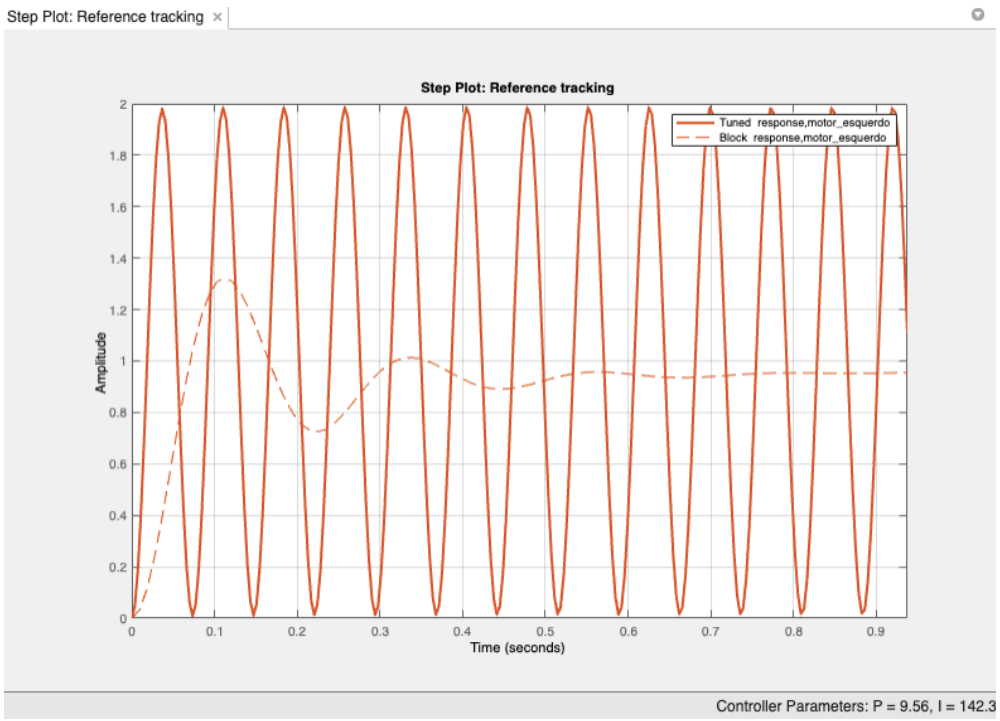
O Gráfico 3 e o Gráfico 4 destacam o comportamento mais agressivo do sistema de um modo lento e de um modo rápido, respectivamente.

Gráfico 3 – Resposta do sistema associado a um controlador PI (P:0 e I: 0.1094) agressivo com resposta lenta.



Fonte: Autor.

Gráfico 4 – Resposta do sistema associado a um controlador PI (P: 9.56 e I: 142.3) agressivo com resposta rápida.

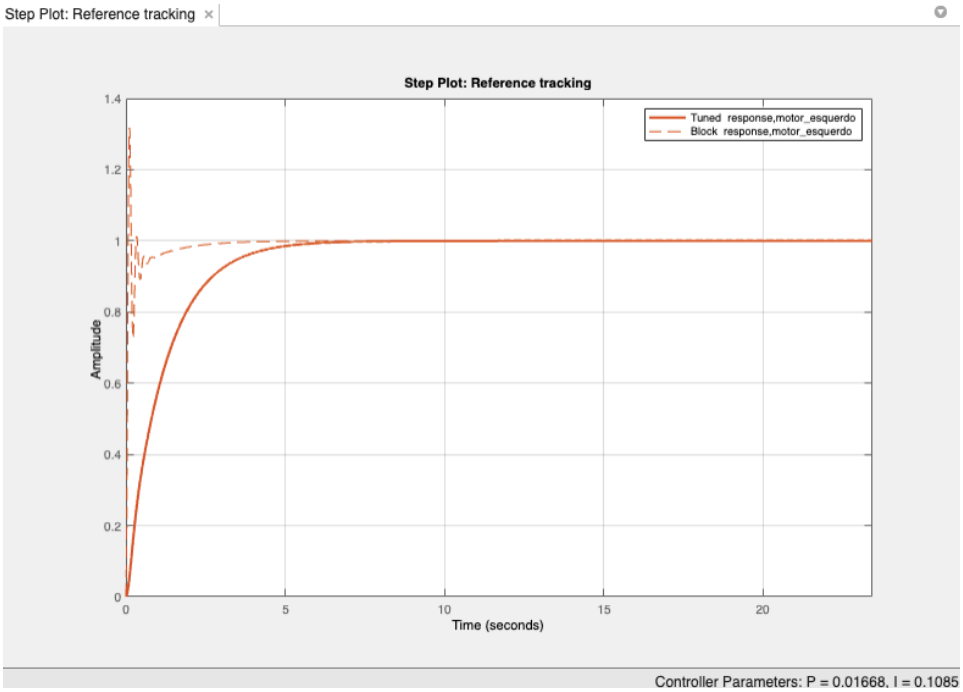


Fonte: Autor.

Paralelamente, também foi possível perceber que que ao manter um comportamento mais robusto do sistema, o novo controlador permite que o Ganho P esteja compreendido dentro de um intervalo de valores entre 0.016679 e 9.7026, enquanto o ganho I esteja compreendido dentro de um intervalo de valores entre 0.10852 e 14.4894.

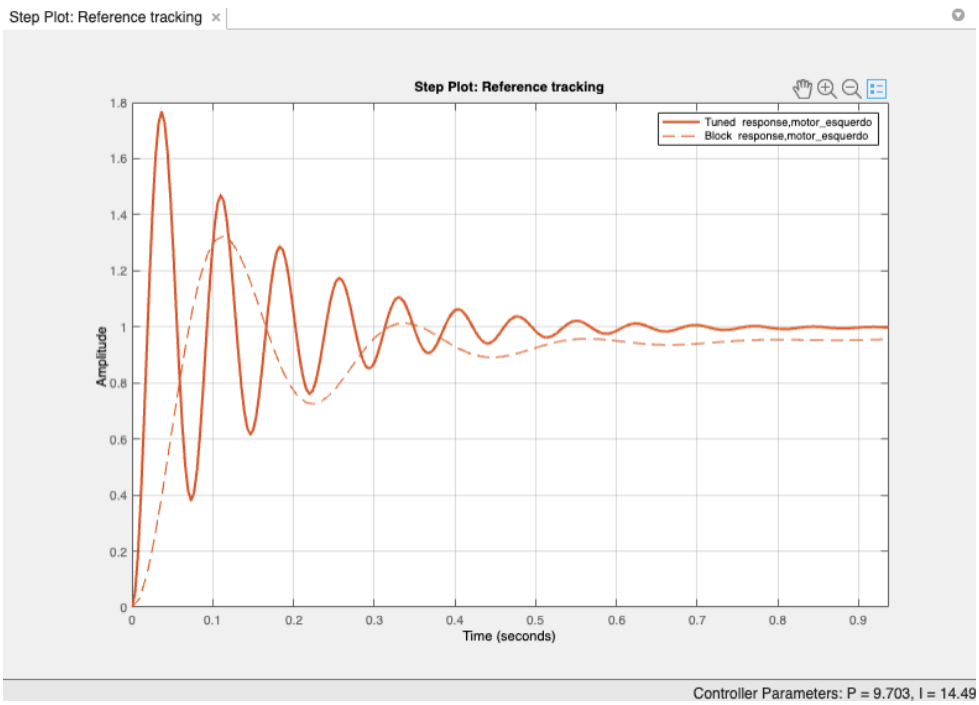
O Gráfico 5 e o Gráfico 6 destacam o comportamento mais robusto do sistema de um modo lento e de um modo rápido, respectivamente.

Gráfico 5 – Resposta do sistema associado a um controlador PI (P: 0.01668 e I: 0.1085) robusto com resposta lenta.



Fonte: Autor.

Gráfico 6 – Resposta do sistema associado a um controlador PI (P: 9.703 e I: 14.49) robusto com resposta rápida.



Fonte: Autor.

Ao avaliar o conjunto de gráficos acima, é evidente que o comportamento entre elas é diferente em certos casos. Principalmente, no quesito relacionado a quantidade de oscilações, ou até mesmo o tempo necessário para a sair do regime transitório e atingir o regime permanente.

Outras diferenças mais específicas podem ser visualizadas nas tabelas abaixo, que abordam informações como o tempo de subida (Rising Time), o tempo de acomodação (Settling Time), o sobressinal (Overshoot), o tempo de pico (Peak), entre outros.

Avaliando os valores assumidos por cada parâmetro na Tabela 18, é interessante destacar que o tempo de subida e o tempo de acomodação aumentaram seus valores significativamente, em relação a faixa observada no Controlador P.

Embora o tempo de pico tenha se mantido próximo aos resultados anteriores, o mais interessante foi o fato de não existir um sobressinal presente na curva de resposta do sistema.

Tabela 18 – Parâmetros controlador PI com resposta agressiva e lenta.

Performance and Robustness

	Tuned
Rise time	2.21 seconds
Settling time	4.1 seconds
Overshoot	0 %
Peak	1
Gain margin	24.7 dB @ 9.81 rad/s
Phase margin	82.5 deg @ 0.853 rad/s
Closed-loop stability	Stable

Fonte: Autor.

Por outro lado, a Tabela 19 deixa evidente que o controlador PI agressivo e rápido, mesmo possuindo estabilidade em malha fechada, tem um comportamento muito distorcido. As oscilações presentes na resposta do sistema se mantem constantes ao longo do tempo, logo não é possível avaliar o tempo de subida, o tempo de acomodação, o sobressinal, ou até mesmo o tempo de pico.

Consequentemente, é uma curva de comportamento extremo, que poderia impactar diretamente no funcionamento do protótipo, caso os ganhos fossem calibrados dessa forma.

Tabela 19 – Parâmetros controlador PI com resposta agressiva e rápida.

Performance and Robustness

	Tuned
Rise time	NaN seconds
Settling time	NaN seconds
Overshoot	NaN %
Peak	Inf
Gain margin	2.7e-05 dB @ 85.3 rad/s
Phase margin	-1.71e-08 deg @ 85.3 rad/s
Closed-loop stability	Stable

Fonte: Autor.

No caso do Controlador PI robusto e lento, conforme ilustrado pela Tabela 20, o comportamento assumido pela resposta do sistema é extremamente semelhante àquele observado na Tabela 18.

As variações no tempo de subida, no tempo de acomodação e no tempo de pico são ínfimas, e outro detalhe importante é fato do sobressinal também não existir nesse caso.

Tabela 20 – Parâmetros controlador PI com resposta robusta e lenta.

Performance and Robustness

	Tuned
Rise time	2.56 seconds
Settling time	4.64 seconds
Overshoot	0 %
Peak	0.998
Gain margin	Inf dB @ Inf rad/s
Phase margin	90 deg @ 0.853 rad/s
Closed-loop stability	Stable

Fonte: Autor.

No caso do Controlador PI robusto e rápido, conforme ilustrado pela Tabela 21, o comportamento assumido pela resposta do sistema possui algumas diferenças significativas em relação aos demais controladores PI.

Por exemplo, o tempo de subida, o tempo de acomodação foram reduzidos, contudo houve o aparecimento de sobressinal, além do tempo de pico ter sofrido uma acentuação em seu valor.

Tabela 21 – Parâmetros controlador PI com resposta robusta e rápida.

Performance and Robustness

	Tuned
Rise time	0.013 seconds
Settling time	0.595 seconds
Overshoot	76.7 %
Peak	1.77
Gain margin	Inf dB @ Inf rad/s
Phase margin	8.89 deg @ 85.3 rad/s
Closed-loop stability	Stable

Fonte: Autor.

Portanto, é possível verificar que o aumento do Ganho proporcional (P) e do ganho Integral (I) tem a capacidade de impactar de maneiras diferentes o comportamento do sistema.

Caso, se mantenha um caráter mais agressivo da resposta, essa elevação dos seus respectivos valores repercute em um aumento exponencial do tempo de subida, do tempo de acomodação, do tempo de pico e do sobressinal. Logo, a resposta tende ao infinito.

Paralelamente, caso se mantenha um caráter mais robusto da reposta, essa elevação dos seus respectivos valores repercute em uma diminuição do tempo de subida e do tempo de acomodação. Mas, acarreta prejuízos ao tempo de pico e ao sobressinal, uma vez que aumenta seus respectivos valores.

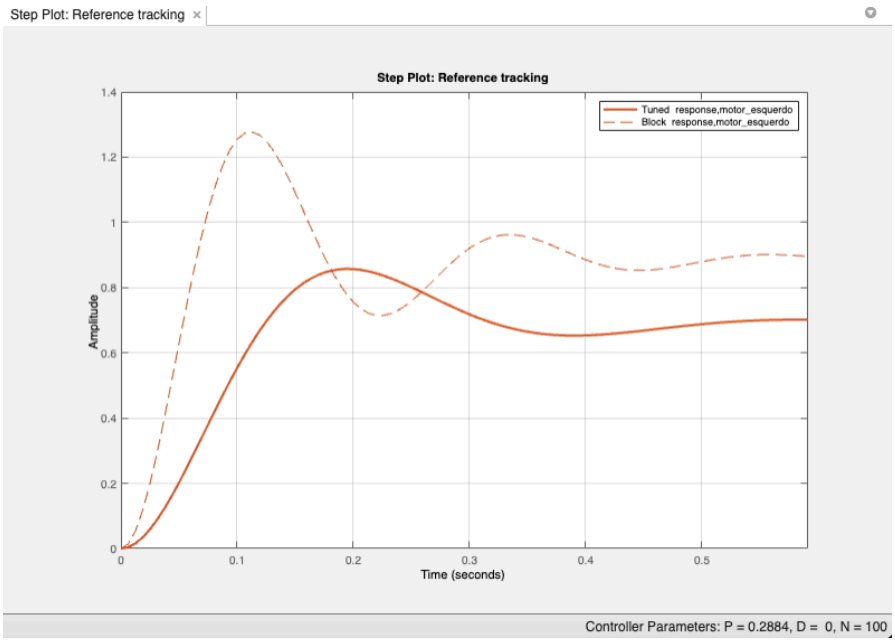
4.1.4 Controlador Proporcional-Derivativo (PD):

No caso do Controlador Proporcional-Derivativo (PD) a faixa de valores determinado pelo uso do “PID tuner” também tem uma interdependência entre os ganhos. Dessa forma, novamente existe uma diferença conforme o Ganho Proporcional é ajustado, assim como o Ganho Derivativo.

Foi possível perceber que ao manter um comportamento mais agressivo do sistema, o novo controlador permite que o Ganho P esteja compreendido dentro de um intervalo de valores entre 0.28885 e 539.4315, enquanto o Ganho D esteja compreendido dentro de um intervalo de valores entre 0 e 1.7404.

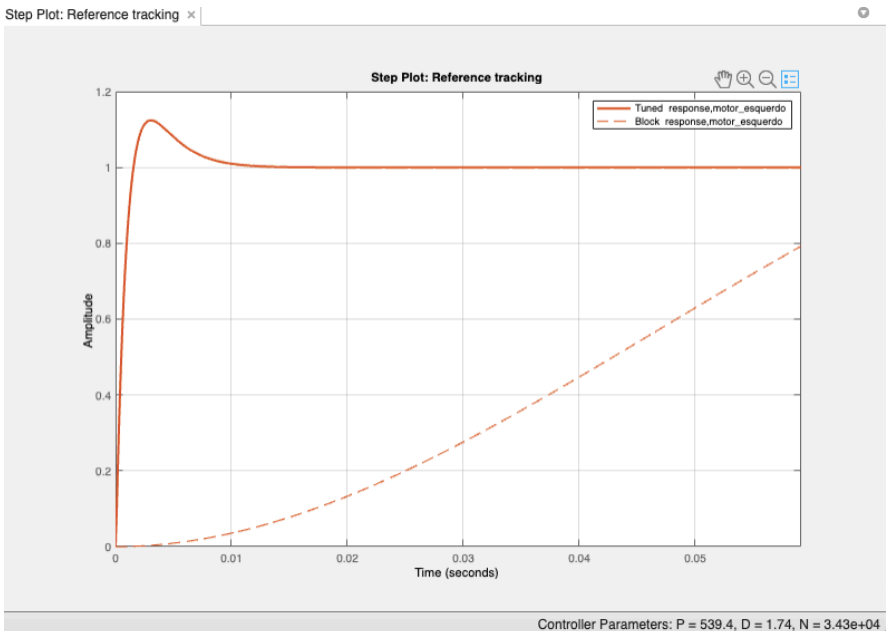
O Gráfico 7 e o Gráfico 8 destacam o comportamento mais agressivo do sistema de um modo lento e de um modo rápido, respectivamente.

Gráfico 7 – Resposta do sistema associado a um controlador PD (P: 0.2884 e D: 0) agressivo com resposta lenta.



Fonte: Autor.

Gráfico 8 – Resposta do sistema associado a um controlador PD (P: 539.4 e D: 1.74) agressivo com resposta rápida.

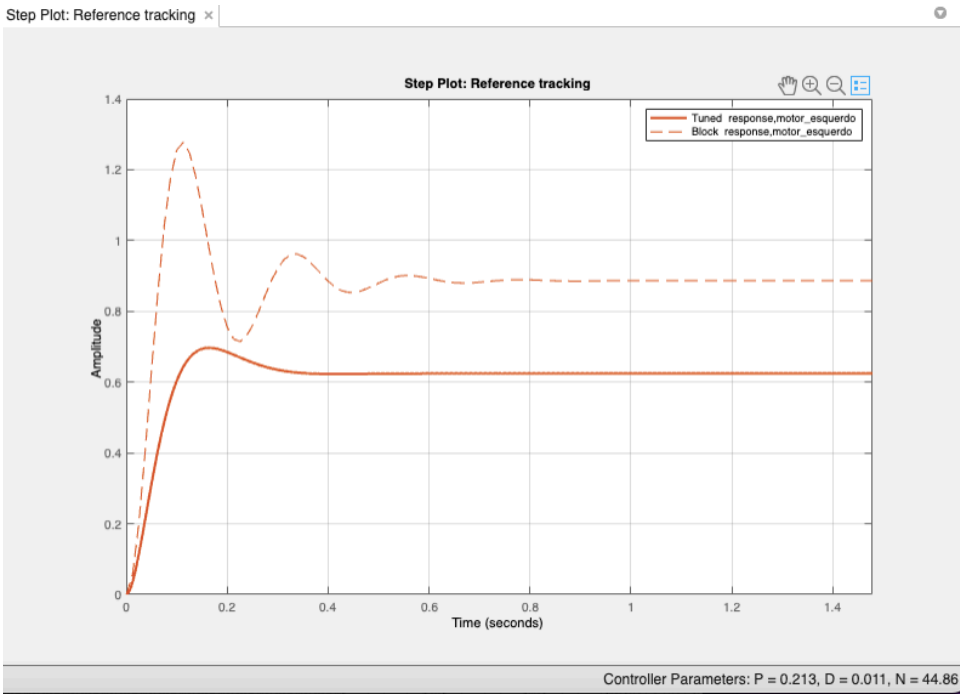


Fonte: Autor.

Paralelamente, foi possível perceber que que ao manter um comportamento mais robusto do sistema, o novo controlador permite que o Ganho P esteja compreendido dentro de um intervalo de valores entre 0.21296 e 21.2936, enquanto o Ganho D esteja compreendido dentro de um intervalo de valores entre 0.011 e 1.7994.

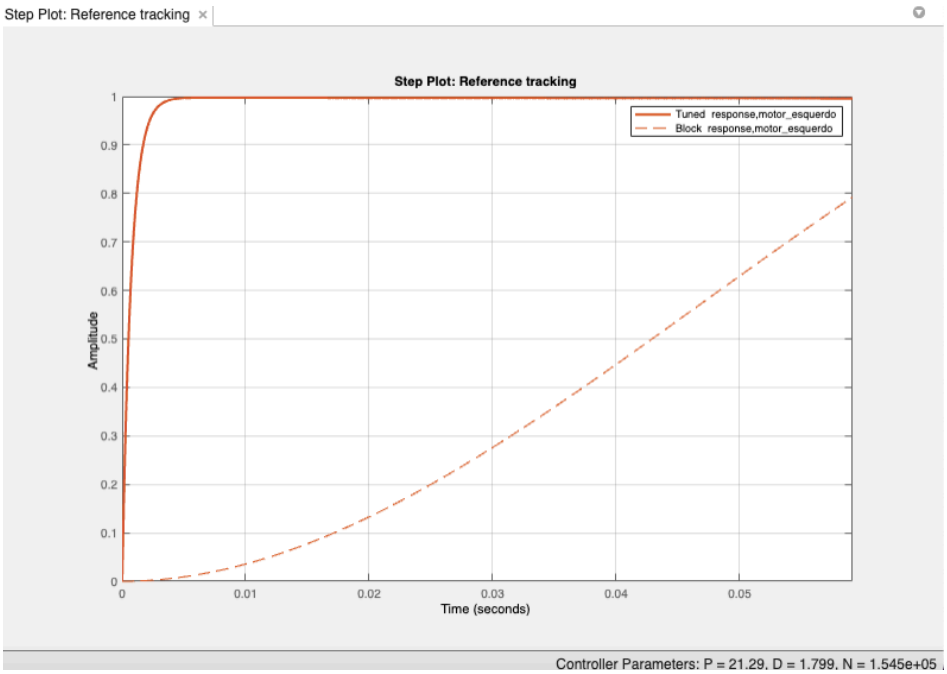
O Gráfico 9 e o Gráfico 10 destacam o comportamento mais robusto do sistema de um modo lento e de um modo rápido, respectivamente.

Gráfico 9 – Resposta do sistema associado a um controlador PD (P: 0.213 e D: 0.011) robusto com resposta lenta.



Fonte: Autor.

Gráfico 10 - Resposta do sistema associado a um controlador PD (P: 21.29 e D: 1.799) robusto com resposta rápida.



Fonte: Autor.

O conjunto de gráficos acima, novamente ressalta que o comportamento entre eles é diferente. No caso do Controlador PD, já não existe um número significativo de oscilações presente na curva de resposta do sistema, seja ela agressiva (lenta ou rápida), seja ela robusta (lenta ou rápida).

Outro detalhe interessante é a passagem do regime transitório para o regime permanente ocorrer de uma maneira mais nítida e suave em relação aos controladores anteriores.

Como nos outros casos, as diferenças mais específicas podem ser visualizadas nas tabelas abaixo, que abordam informações como o tempo de subida (Rising Time), o tempo de acomodação (Settling Time), o sobressinal (Overshoot), o tempo de pico (Peak), entre outros.

Na Tabela 22 é possível verificar que no caso do Controlador PD, o tempo de subida, o tempo de acomodação e o tempo de pico não são muito elevados, como visto no Controlador PI, eles assumem uma faixa de valores mais próxima do Controlador P. Também é possível observar que o máximo sobressinal está presente em uma faixa mais contida, se comparado as outras situações.

Tabela 22 – Parâmetros controlador PD com resposta agressiva e lenta.

Performance and Robustness

	Tuned
Rise time	0.0842 seconds
Settling time	0.475 seconds
Overshoot	23.8 %
Peak	0.857
Gain margin	Inf dB @ Inf rad/s
Phase margin	66.5 deg @ 13.5 rad/s
Closed-loop stability	Stable

Fonte: Autor.

Por outro lado, a Tabela 23 deixa evidente que o controlador PD agressivo e rápido, possui um tempo de subida e um tempo de acomodação ainda menores. O máximo sobressinal, continua existindo, mas foi levemente reduzido, assim como o tempo de pico.

Tabela 23 – Parâmetros controlador PD com resposta agressiva e rápida.

Performance and Robustness

	Tuned
Rise time	0.00109 seconds
Settling time	0.00844 seconds
Overshoot	12.4 %
Peak	1.12
Gain margin	Inf dB @ Inf rad/s
Phase margin	75.6 deg @ 1.35e+03 rad/s
Closed-loop stability	Stable

Fonte: Autor.

No caso do controlador PD robusto e lento, conforme ilustrado pela Tabela 24, o comportamento assumido pela resposta do sistema é extremamente semelhante àquele observado no controlador PD agressivo e lento. Logo, se aproxima do Controlador P e de algumas modalidades do Controlador PI.

As reduções no tempo de subida e no tempo de acomodação não são muito grandes. O máximo sobressinal e o tempo de pico são os parâmetros que mais sofreram uma diminuição significativa.

Tabela 24 – Parâmetros controlador PD com resposta robusta e lenta.

Performance and Robustness

	Tuned
Rise time	0.0723 seconds
Settling time	0.293 seconds
Overshoot	11.6 %
Peak	0.697
Gain margin	Inf dB @ Inf rad/s
Phase margin	94.7 deg @ 13.5 rad/s
Closed-loop stability	Stable

Fonte: Autor.

No caso do controlador PD robusto e rápido, conforme ilustrado pela Tabela 25, o comportamento assumido pela resposta do sistema possui o menor tempo de subida e acomodação, além de contar com um máximo sobressinal extremamente baixo. Mas, seu tempo de pico ainda é lento se comparado aos outros controladores PD.

Tabela 25 – Parâmetros controlador PD com resposta robusta e rápida.

Performance and Robustness

	Tuned
Rise time	0.00158 seconds
Settling time	0.00274 seconds
Overshoot	0.305 %
Peak	0.997
Gain margin	Inf dB @ Inf rad/s
Phase margin	89.6 deg @ 1.35e+03 rad/s
Closed-loop stability	Stable

Fonte: Autor.

Portanto, é possível verificar que o aumento do ganho proporcional (P) e do ganho derivativo (D) tem a capacidade de impactar de maneiras diferentes o comportamento do sistema. Entretanto, essa correlação existente possui algumas mudanças se comparado ao caso anterior, envolvendo o ganho proporcional (P) e o ganho integral (I).

Caso, se mantenha um caráter mais agressivo da resposta, essa elevação dos seus respectivos valores repercute em uma diminuição do tempo de subida, do tempo de acomodação e do sobressinal. Mas, culmina no aumento do tempo de pico.

Paralelamente, caso se mantenha um caráter mais robusto da reposta, essa elevação dos seus respectivos valores repercute em uma diminuição ainda mais significativa do tempo de subida e do tempo de acomodação. O sobressinal segue o mesmo caminho sofrendo uma redução a ponto de se tornar insignificante. Embora ainda acarrete prejuízos ao tempo de pico, ocasionando uma elevação moderada.

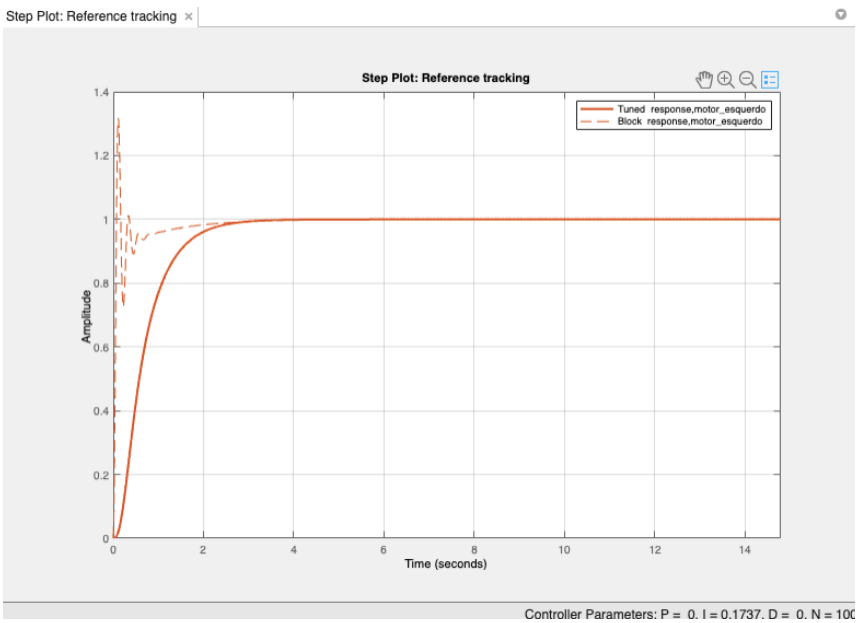
4.1.5 Controlador Proporcional-Integral-Derivativo (PID):

No caso do Controlador Proporcional-Integral-Derivativo (PID) a faixa de valores determinada pelo uso do "PID tuner" apresenta as mesmas particularidades em relação ao comportamento dos ganhos vistos nos controladores anteriores. O termo proporcional, o termo integral e o termo derivativo sofrem alterações conforme seus respectivos ganhos são ajustados.

Foi possível perceber que ao manter um comportamento mais agressivo do sistema, o novo controlador permite que o ganho P esteja compreendido dentro de um intervalo de valores entre 0 e 21.191, enquanto o ganho I esteja compreendido dentro de um intervalo de valores entre 0.17372 e 264.6836, e o ganho D dentro de um intervalo de valores entre 0 e 0.046309.

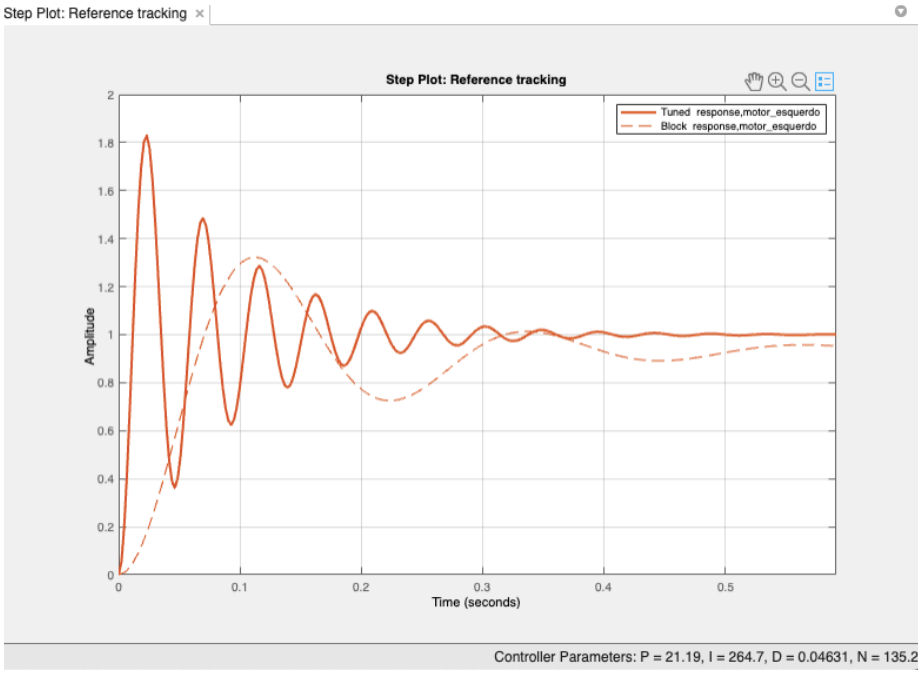
O Gráfico 11 e o Gráfico 12 destacam o comportamento mais agressivo do sistema de um modo lento e de um modo rápido, respectivamente.

Gráfico 11 – Resposta do sistema associado a um controlador PID (P: 0.1, I: 0.1737 e D: 0) agressivo com resposta lenta.



Fonte: Autor.

Gráfico 12 – Resposta do sistema associado a um controlador PID (P: 21.19, I: 264.7 e D: 0.04631) agressivo com resposta rápida.

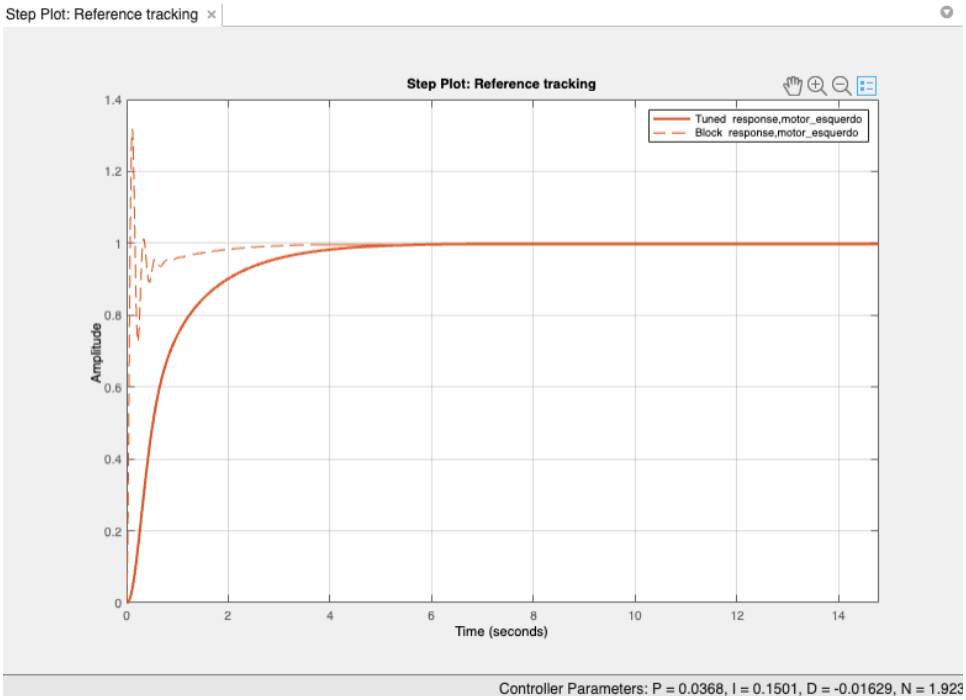


Fonte: Autor.

Paralelamente, também foi possível perceber que que ao manter um comportamento mais robusto do sistema, o novo controlador permite que o Ganho P esteja compreendido dentro de um intervalo de valores entre 0.036796 e 2.431, enquanto o Ganho I esteja compreendido dentro de um intervalo de valores entre 0.15007 e 2.626, e o Ganho D compreendido dentro de um intervalo de valores entre 0.026286 e 0.1792.

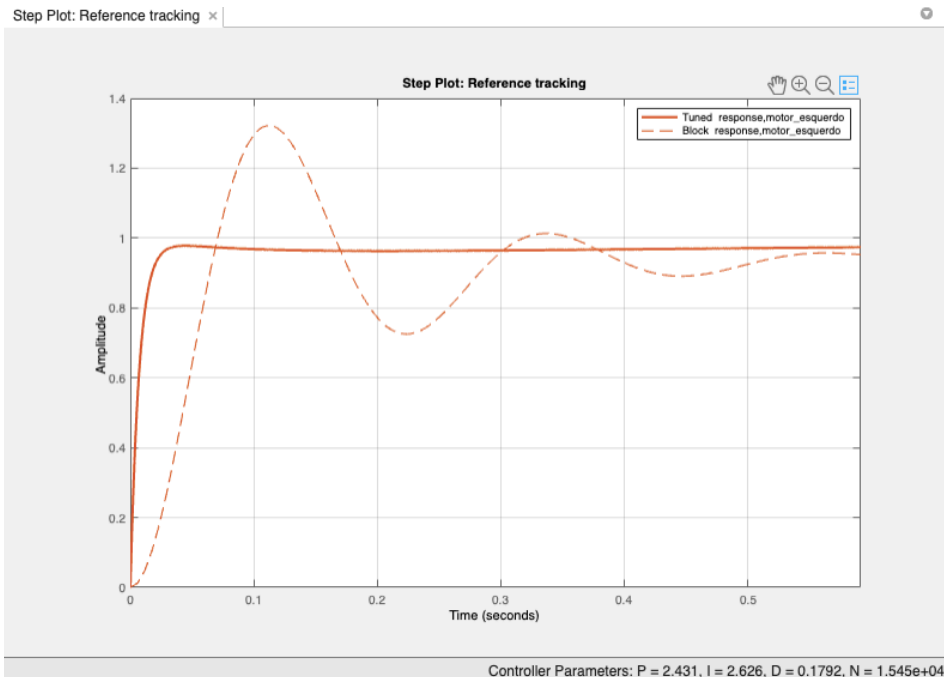
O Gráfico 13 e o Gráfico 14 destacam o comportamento mais robusto do sistema de um modo lento e de um modo rápido, respectivamente.

Gráfico 13 – Resposta do sistema associado a um controlador PID (P: 0.0368, I: 0.1501 e D: -0.01629) robusto com resposta lenta.



Fonte: Autor.

Gráfico 14 – Resposta do sistema associado a um controlador PID (P: 2.431, I: 2.626 e D: 0.1792) robusto com resposta rápida.



Fonte: Autor.

Ao avaliar o conjunto de gráficos acima, é evidente que o comportamento entre elas é bastante semelhante. As respostas do Controlador PID calibrado com ganhos distintos, e consequentemente diferentes valores não desencadeou uma grande variedade de curvas de resposta.

Mais precisamente, as curvas, em sua grande maioria não apresentaram muitas oscilações, como também possuem uma transição suave entre o regime transitório e o regime permanente.

Outras diferenças mais específicas podem ser visualizadas nas tabelas abaixo, que abordam informações como o tempo de subida (Rising Time), o tempo de acomodação (Settling Time), o sobressinal (Overshoot), o tempo de pico (Peak), entre outros.

Avaliando os valores assumidos por cada parâmetro na Tabela 26, é interessante destacar que o tempo de subida e o tempo de acomodação estão próximos da faixa de valores já observada em controladores anteriores. O tempo de pico se manteve baixo, e o sobressinal é inexistente, assim como em outros casos.

Tabela 26 – Parâmetros controlador PID com resposta agressiva e lenta.

Performance and Robustness

	Tuned
Rise time	1.25 seconds
Settling time	2.37 seconds
Overshoot	0 %
Peak	1
Gain margin	20.7 dB @ 9.81 rad/s
Phase margin	78.1 deg @ 1.35 rad/s
Closed-loop stability	Stable

Fonte: Autor.

Por outro lado, a Tabela 27 deixa evidente que o controlador PID agressivo e rápido possui um tempo de subida e um tempo de acomodação menores. Entretanto, é possível constatar que tempo de pico e o máximo sobressinal assumem valores significativos, além de elevados.

Tabela 27 – Parâmetros controlador PID com resposta agressiva e rápida.

Performance and Robustness	
	Tuned
Rise time	0.00782 seconds
Settling time	0.331 seconds
Overshoot	82.9 %
Peak	1.83
Gain margin	Inf dB @ Inf rad/s
Phase margin	9 deg @ 135 rad/s
Closed-loop stability	Stable

Fonte: Autor.

No caso do controlador PID robusto e lento, conforme ilustrado pela Tabela (28), o comportamento da curva de resposta do sistema é semelhante àquele observado no controlador PID agressivo e lento.

As variações no tempo de subida, no tempo de acomodação e no tempo de pico são ínfimas, e outro detalhe importante é fato do máximo sobressinal também não existir nesse caso.

Tabela 28 – Parâmetros controlador PID com resposta robusta e lenta.

Performance and Robustness	
	Tuned
Rise time	1.81 seconds
Settling time	3.85 seconds
Overshoot	0 %
Peak	0.999
Gain margin	24 dB @ 13 rad/s
Phase margin	90 deg @ 1.35 rad/s
Closed-loop stability	Stable

Fonte: Autor.

No caso do controlador PID robusto e rápido, conforme ilustrado pela Tabela 29, o comportamento assumido pela resposta do sistema possui algumas diferenças significativas em relação aos demais controladores PID.

Por mais que o tempo de subida, o tempo de pico e sobressinal estejam dentro da faixa de valores presente em outros controladores, o tempo de acomodação não foi computado.

Tabela 29 – Parâmetros controlador PID com resposta robusta e rápida.

Performance and Robustness

	Tuned
Rise time	0.0168 seconds
Settling time	NaN seconds
Overshoot	0 %
Peak	0.978
Gain margin	Inf dB @ Inf rad/s
Phase margin	90 deg @ 135 rad/s
Closed-loop stability	Stable

Fonte: Autor.

Portanto, é possível verificar que o aumento do ganho Proporcional (P), do ganho Integral (I) e do ganho Derivativo (D) tem a capacidade de impactar de maneiras diferentes o comportamento do sistema.

Entretanto, essa correlação existente possui algumas mudanças ao comparar com os casos anteriores, uma vez que os controladores PID contam com a presença das três modalidades de ganho.

Caso, se mantenha um caráter mais agressivo da resposta, essa elevação dos seus respectivos valores repercute em uma diminuição do tempo de subida e do tempo de acomodação, mas culmina no aumento do tempo de pico e do sobressinal.

Paralelamente, caso se mantenha um caráter mais robusto da reposta, essa elevação dos seus respectivos valores repercute em uma diminuição ainda mais significativa do tempo de subida. Em compensação o tempo de pico e o sobressinal não sofreram alterações significativas em seus valores.

Um detalhe importante a se destacar é que nenhuma conclusão pode ser feita em relação ao tempo de acomodação. Mais precisamente, o fato de um valor não ter sido identificado denota a presença de um problema para sair do regime transitório e atingir o regime permanente.

4.2 Resultados dos testes em bancada:

No caso dos testes em bancada empregando o protótipo real não existia a possibilidade de utilizar o recurso do “PID tuner” presente no Matlab por meio do SIMULINK. Como alternativa os dados e as informações coletadas por meio do desenvolvimento dos códigos no Visual Studio Code foram utilizados na forma de elementos base para observação de tendências.

Ao final de cada simulação realizada, os dados referentes ao robô eram armazenados em tabelas no formato (.csv) construídas em tempo real com o uso das ferramentas de "Datalog". As tabelas dos controladores ON/OFF, de 3 Níveis e Proporcional (P), computavam o tempo decorrido, o erro associado a cada leitura e o nível de curva correspondente.

Enquanto, o Controlador Proporcional-Integral (PI) e o Controlador Proporcional-Derivativo (PD) computavam também o valor do termo integrativo e do termo derivativo, respectivamente. Por último o Controlador Proporcional-Integral-Derivativo (PID) computava todos os valores citados anteriormente.

Uma parte fundamental dessa análise quantitativa está baseada no cálculo da dispersão. Ela é uma medida estatística responsável por avaliar a distribuição de um conjunto de informações. Nessa situação, ela foi empregada para avaliar o comportamento do erro característico de cada controlador. Sendo assim, permite determinar o quanto cada um dos valores de erro está mais próximo, ou mais distante da média aritmética. Logo, a cada conjunto de ganhos implementados nos controladores, se tornou possível verificar se os valores de erro estão mais concentrados em torno da média, ou mais espalhados. Entretanto, como o existe um certo ruído inerente do sistema sobre os erros obtidos, o cálculo da média acaba ficando suscetível a uma certa flutuação devido a presença de amostras onde o ruído é significativo.

Portanto, a análise quantitativa também conta com o cálculo do erro quadrático médio (EQM). Ele é outra medida estatística responsável por avaliar o grau de precisão (acurácia) do controlador utilizando como parâmetro os valores de erro característico.

O erro quadrático médio também possui outras capacidades mais especificas, o que possibilitou observar outras características dos controladores, principalmente a equiparação entre o comportamento do robô em função do erro advindo da implementação lógica do controlador por meio do código.

A análise dessas estatísticas desempenhou um papel fundamental na quantificação do desempenho do robô, permitindo estabelecer comparações significativas entre os controladores e avaliar sua eficácia, fornecendo uma base sólida para as conclusões e recomendações.

4.2.1 Controlador ON/OFF e Controlador de 3 Níveis:

No caso do controlador ON/OFF, que possui um comportamento bastante simples, e consequentemente tanto na simulação virtual, quanto na simulação real, foi possível identificar que o seu desempenho é o pior em relação aos demais.

Mais precisamente, o controlador ON/OFF possui apenas duas referências para o desenvolvimento do trajeto em questão, isto é, virar à esquerda, quando o valor é inferior a 27,5%, ou virar à direita, quando o valor é superior a 27,5%.

Por outro lado, o controlador de 3 Níveis possui a capacidade de realizar mais uma ação, isto é, além de virar à esquerda e à direita, o robô também pode ir reto. Essa nova ação torna esse controlador um aprimoramento do controlador ON/OFF.

Desse modo, a cada nova leitura um dos motores recebe uma porcentagem de tensão elétrica, enquanto no outro o fornecimento é cessado totalmente. Essa sucessão de estados, nível lógico alto e nível lógico baixo, para cada um dos motores cria uma oscilação no comportamento do robô, fazendo com que ele execute uma espécie de zigue-zague, enquanto avança.

Como existem apenas duas possibilidades de movimento, o robô não tem a capacidade de fazer ajustes acurados e precisos em relação a trajetória a ser percorrida. Quando, o erro é contabilizado nas leituras, a compensação feita pelo sistema de controle acaba se sobressaindo em relação a real necessidade.

O resultado é uma atuação exagerada dos motores, e mesmo sendo capaz de completar o percurso, existe uma diferença de desempenho nas seções retilíneas e nas seções curvilíneas. Logo, fica abaixo do esperado, pois ele não segue a linha de uma maneira uniforme.

É possível observar o ideal de não holonomia muito presente em seu comportamento, uma vez que uma curva ao ser realizada é decomposta em uma série de movimentos retilíneos que buscam se aproximar de seu formato.

As tabelas seguintes apresentam os resultados encontrados para o desvio padrão e o erro quadrático médio associado a esses controladores.

Tabela 30 – Dispersão controlador ON/OFF.

CONTROLADOR ON/OFF	
MÉDIA	-4,278947368
DISPERSÃO	199,4958507
SOMA (CURVA)	37704,71579
OSCILACÃO DO ROBÔ	GRANDE

Fonte: Autor.

Tabela 31 – Erro quadrático médio controlador ON/OFF.

CONTROLADOR ON/OFF	
EQM(SOMA/N)	215,6204188

Fonte: Autor.

Tabela 32 – Dispersão controlador de 3 Níveis.

CONTROLADOR DE 3 NÍVEIS	
MÉDIA	0,76036036
DISPERSÃO	54,17532117
SOMA (CURVA)	30013,12793
OSCILACÃO DO ROBÔ	MÉDIA

Fonte: Autor.

Tabela 33 – Erro quadrático médio controlador de 3 Níveis.

CONTROLADOR 3 NÍVEIS	
EQM(SOMA/N)	54,55755396

Fonte: Autor.

Analisando os resultados presentes na Tabela 30 e na Tabela 32 é evidente que o Controlador de 3 Níveis possui um desempenho melhor em relação ao Controlador ON/OFF.

Basicamente, o erro característico do Controlador ON/OFF apresenta uma maior dispersão, resultando em uma maior oscilação do robô ao longo do trajeto. Já o erro característico do controlador de 3 Níveis apresenta uma dispersão menor, resultando em uma menor oscilação do robô ao longo do trajeto.

O parâmetro soma também revela um detalhe interessante, uma vez que contabiliza a soma de todos os valores de potência (porcentagem) enviados aos motores durante a execução do percurso. Desse modo, como o Controlador ON/OFF tem um maior valor, ele acabou necessitando de um maior nível de energia para executar o mesmo trajeto do controlador de 3 Níveis.

A mesma situação é verificada por meio da comparação entre a Tabela 31 e a Tabela 33, que contabilizam o erro quadrático médio dos dois controladores. Sendo assim, novamente o controlador ON/OFF possui um erro quadrático médio superior ao do controlador de 3 Níveis.

Ou seja, o Controlador de 3 Níveis é mais preciso (acurado) em relação ao Controlador ON/OFF. Portanto, é válido afirmar que o controlador de 3 Níveis tem um desempenho superior ao controlador ON/OFF.

Contudo, ao longo dos testes foi verificado que o Controlador de 3 Níveis por muitas vezes perdeu a borda da linha como referência, e acabou se deslocando sobre o centro da fita isolante. Esse mesmo fenômeno não pode ser observado no Controlador ON/OFF, que permaneceu utilizando a borda da linha como referência durante todo o percurso.

4.2.2 Controlador Proporcional (P):

A simulação realizada no protótipo permitiu perceber que a faixa de ganhos aceitos no caso do Controlador Proporcional era relativamente diferente daquela constatada no ambiente virtual do SIMULINK. Mais precisamente, o ganho proporcional estava concentrado em uma faixa de 2,5 até 10.

Valores inferiores a 2,5 fizeram com que o protótipo não tivesse a capacidade de reconhecer a diferença de intensidade entre a coloração preta e a coloração branca. Desse modo, ele acaba impossibilitado o robô de seguir a linha do trajeto.

Já valores superiores a 10 tiveram o efeito contrário. Eles fizeram com que o protótipo tivesse uma capacidade extremamente exagerada de reconhecer a diferença de intensidade entre a coloração preta e a coloração branca.

O uso de tais valores acabavam impossibilitado de seguir a linha do trajeto, pois a cada nova amostra, o erro obtido ocasionava uma compensação acima do necessário. A tabela abaixo retrata os resultados obtidos para o controlador P em termos da dispersão:

Tabela 34 – Dispersão associada a cada valor de ganho do controlador P.

CONTROLADOR P				
GANHO	**MÉDIA**	**DISPERSÃO (S^2)**	**SOMA (CURVA)**	**OSILAÇÃO DO ROBÔ**
2,5	-10,06382979	62,27854987	14886	PEQUENA
3	-8,733103448	50,38454868	13736	PEQUENA
3,5	-7,16575902	37,5523758	11111,5	PEQUENA
4	-6,360099929	30,07184256	37938	PEQUENA
4,5	-5,698499318	30,02678273	42340,5	PEQUENA
5	-5,130390144	20,8591854	40467,5	MÉDIA
5,5	-4,471409122	18,79618473	41038,25	MÉDIA
6	-4,160324172	18,63912288	40755	MÉDIA
6,5	-3,801360544	17,74233988	42802,5	MÉDIA
7	-3,487188132	20,05719067	46833,5	MÉDIA
7,5	-3,218856365	16,08779815	44403,75	MÉDIA
8	-2,980565371	20,7222175	46900	GRANDE
8,5	-2,807371349	15,20121215	45798	GRANDE
9	-2,794968987	15,83167007	48955,5	GRANDE
9,5	-2,572952512	15,21560016	49765,75	GRANDE
10	-2,49084507	17,41047957	50710	GRANDE

Fonte: Autor.

Analisando os resultados presentes na tabela acima é evidente que existe uma relação entre o aumento do ganho proporcional, a dispersão do erro característico associado a cada valor, como também a soma e a oscilação do robô.

Conforme o ganho proporcional foi aumentado, a dispersão sofreu uma redução. A redução assumiu um comportamento muito próximo de uma linearidade, uma vez que o decrescimento possui até certo ponto uma uniformidade. Somente nos valores finais, isto é, entre 8 e 10, ela é quebrada. Logo, quanto maior o ganho dentro desse intervalo de valores, menor é a dispersão.

Em relação aos valores de potência (porcentagem) contabilizados na coluna curva, a mesma constatação não pode ser feita, uma vez que o aumento do ganho, resultou em um aumento da energia dispendida para a realização do trajeto.

Inicialmente, esse pensamento pode parecer estranho, uma vez que a redução da dispersão, deveria culminar em um melhor desempenho para executar o trajeto. Entretanto, a situação é mais complexa, pois o aumento do ganho também resulta em um aumento no número de amostras coletadas.

Na tabela abaixo, a quantidade de amostras válidas obtidas com a utilização de seu respectivo ganho é destacada:

Tabela 35 – Número de amostras associada a cada valor de ganho do controlador P.

GANHO	AMOSTRAS
2,5	1411
3	1451
3,5	1470
4	1402
4,5	1467
5	1462
5,5	1470
6	1420
6,5	1471
7	1484
7,5	1470
8	1416
8,5	1439
9	1452
9,5	1454
10	1421

Fonte: Autor.

Mais precisamente, mesmo estando estabelecido um período de 10 milissegundos entre as amostras, o hardware teve uma grande dificuldade de manter esse intervalo de tempo ao longo da montagem das tabelas de dados. Consequentemente, existiu uma certa variação no intervalo de tempo.

Essa variação se mantém estável na maior parte das situações, porém repercutiu na diferente quantidade de amostras computadas para cada ganho. Como a dispersão depende da quantidade de amostras para cada valor de ganho, cada vez que o ganho era aumentado, o robô acabava coletando mais dados, fazendo com que ele levasse mais tempo para executar o trajeto.

Portanto, quanto maior o ganho, maior o gasto energético do robô, assim como sua oscilação. O ganho mais alto também influência a capacidade do robô se locomover, e quanto mais próximo os valores estavam do limite, ou seja, do valor 10, existia um aumento da possibilidade de o robô realizar uma compensação exagerada.

Sendo assim, resumidamente, quanto maior o ganho, menor a dispersão, maior o gasto energético e maior a oscilação. Logo, não existe um ponto ótimo, mas sim uma faixa de valores onde é possível fazer com que o desempenho do robô seja mais preciso.

A tabela abaixo reforça essa ideia ao destacar os valores do erro quadrático médio característicos de cada ganho proporcional:

Tabela 36 - Erro quadrático médio associado a cada valor de ganho do controlador P.

CONTROLADOR P	
GANHO	**EQM(SOMA/N)**
2,5	163,4433026
3	126,5296347
3,5	88,81445578
4	70,45096291
4,5	62,4366053
5	47,13354993
5,5	38,7505102
6	35,90897887
6,5	32,15873555
7	32,18244609
7,5	26,41989796
8	29,57044492
8,5	23,05594163
9	23,62723829
9,5	21,82066713
10	23,5981703

Fonte: Autor.

O erro quadrático médio segue uma relação semelhante àquela observada para a dispersão, ou seja, quanto maior o valor do ganho, menor é o erro quadrático médio. Dessa maneira, o padrão de linearidade possui sua uniformidade quebrada novamente entre os valores de ganho equivalentes a 8 e a 10.

Mesmo os ganhos mais altos apresentando EQMs mais baixos, novamente é interessante destacar que a precisão desses controladores é mais alta, visto que possuem uma quantidade de amostras mais elevada, facilitando a manutenção de sua posição em relação a borda de referência. Em contrapartida, o gasto energético é maior, assim como a oscilação.

Sendo assim, o intervalo de 5 a 7,5 se estabelece como a melhor opção para uma boa manutenção da dispersão, do gasto energético e da oscilação, resultando em uma precisão dentro do esperado.

4.2.3 Controlador Proporcional-Integral (PI):

A calibração do ganho desse controlador acabou ocorrendo de um modo diferente da utilizada na simulação virtual, pois a simulação realizada no protótipo exigia em primeiro plano a definição do ganho proporcional. Somente com o ganho proporcional definido, era possível conduzir os procedimentos de ajuste para o ganho integral.

Sendo assim, o ganho proporcional utilizado foi equivalente a 6. Com esse valor previamente definido foi necessário encontrar uma faixa de valores do ganho integrativo onde o funcionamento permitiria o comportamento adequado.

É importante ressaltar novamente que o Controlador PI implementado no protótipo é diferente de um Controlador PI padrão. Mais precisamente, o Controlador PI padrão apresentou problemas na contabilização do erro, uma vez que sem as alterações feitas nas linhas de código, existia um acúmulo excessivo dos valores fazendo com que o termo integral tendesse ao infinito.

A solução encontrada foi criar uma forma de atenuar o erro característico do termo integral, como também impedir a sua mudança de sinal. Desse modo, é possível dizer que o Controlador PI implementado é uma versão adaptada.

Com os testes executados a faixa de ganho aceita pelo termo integral estava compreendida entre 0 e 1. Acima de 1 o robô tinha um comportamento exagerado, e acabava executando movimentos extremamente abruptos. Logo, era impossível conduzir o seguimento do trajeto.

Valores abaixo de 0 não foram testados, uma vez que o comportamento do erro acabava mudando de sinal. Como consequência uma nova alteração do código seria necessária para compensar esses valores. Sendo assim, foi mais interessante manter uma uniformidade no sinal que acompanhava os valores de erro.

A tabela abaixo retrata os resultados obtidos para o controlador PI em termos da dispersão:

Tabela 37 – Dispersão associada a cada valor de ganho do controlador PI.

CONTROLADOR PI				
GANHO	**MÉDIA**	**DISPERSÃO (S^2)**	**SOMA (CURVA)**	**OSILAÇÃO DO ROBÔ**
1	-6,727979275	28,42502399	39745,36323	MÉDIA
0,1	-4,465542522	18,35464453	39907,57692	MÉDIA
0,01	-4,395772595	21,79875673	42105,08679	PEQUENA
0,001	-4,3368846	19,19749331	41276,35376	PEQUENA
0,0001	-4,352422907	21,1008348	41722,73294	PEQUENA

Fonte: Autor.

Analisando os resultados presentes na tabela acima é evidente que existe uma relação entre a diminuição do ganho integral, a dispersão do erro característico associado a cada valor, como também a soma e a oscilação do robô.

Conforme o ganho integral foi reduzido, a dispersão sofreu uma diminuição também. Essa diminuição assumiu um comportamento próprio. Não foi possível determinar a existência de uma uniformidade, contudo o decrescimento sofreu uma queda levemente acentuada entre 1 e 0.1, e posteriormente assumiu uma característica oscilatória entre 0.1 e 0.0001. Logo, quanto menor o ganho dentro desse intervalo de valores, menor é a dispersão também.

Em relação aos valores de potência (porcentagem) contabilizados na coluna curva, a mesma constatação não pode ser feita, uma vez que a diminuição do ganho até a faixa de 0.01, resultou em um aumento da energia dispendida para a realização do trajeto. Contudo, posteriormente, uma pequena queda é visível, se assemelhando ao comportamento constante observado entre 0.01 e 0.0001.

Logo, conforme o ganho é reduzido, inicialmente o gasto energético aumenta. Posteriormente, a diminuição perde influência e o gasto energético passa a oscilar dentro de uma faixa contínua.

Esse pensamento pode parecer estranho, uma vez que a redução da dispersão, deveria culminar em um melhor desempenho para executar o trajeto. Entretanto, a situação é mais complexa, pois a diminuição do ganho também resulta em variação no número de amostras coletadas.

Na tabela abaixo, a quantidade de amostras válidas obtidas com a utilização de seu respectivo ganho é destacada:

Tabela 38 – Número de amostras associadas a cada valor de ganho do controlador PI.

CONTROLADOR PI	
GANHO	**AMOSTRAS**
1	1352
0,1	1365
0,01	1373
0,001	1362
0,0001	1364

Fonte: Autor.

Mesmo estando estabelecido um período de 10 milissegundos entre as amostras, o hardware teve uma grande dificuldade de manter esse intervalo de tempo ao longo da montagem das tabelas de dados. Consequentemente, existiu uma certa variação no intervalo de tempo.

Essa variação se mantém estável na maior parte das situações, porém repercutiu na quantidade de amostras diferente computadas para cada ganho, como no caso do controlador anterior.

Como a dispersão depende da quantidade de amostras para cada valor de ganho, cada vez que o ganho era reduzido, o robô tinha uma tendência de coletar mais dados, fazendo com que ele levasse mais tempo para executar o trajeto.

Entretanto, como destacado no caso do gasto energético, chega um momento em que a redução não causa mais um efeito significativo, e as amostras também ficam em torno de um número constante. Portanto, quanto menor o ganho, o gasto energético do robô assume um comportamento oscilatório próprio, enquanto a sua oscilação é reduzida.

O ganho mais alto também influência a capacidade do robô se locomover, e quanto mais próximo os valores estavam do limite, ou seja, do valor 1, existia um aumento da possibilidade de o robô realizar uma compensação exagerada, devido a isso a oscilação é mais presente nessa faixa.

Sendo assim, resumidamente, quanto menor o ganho, menor a dispersão até determinado ponto, assim como o gasto energético e até mesmo a oscilação. Depois, a diminuição deixa de ter efeito e os valores se estabilizam dentro de uma faixa específica. Logo, não existe um ponto ótimo, mas sim uma faixa de valores onde é possível fazer com que o desempenho do robô seja mais adequado a necessidade.

A tabela abaixo reforça essa ideia ao destacar os valores do erro quadrático médio característicos de cada ganho integral:

Tabela 39 – Erro quadrático médio associado a cada valor de ganho do controlador PI.

CONTROLADOR PI	
GANHO	**EQM(SOMA/N)**
1	73,6151997
0,1	38,25421245
0,01	41,07574654
0,001	37,97815712
0,0001	40,01504035

Fonte: Autor.

O erro quadrático médio segue uma relação semelhante àquela observada para a dispersão, ou seja, quanto menor o valor do ganho, menor é o erro quadrático médio. Entretanto, isso ocorre até uma certa faixa de valores, posteriormente a redução deixa de ter um efeito e o erro quadrático médio passa a oscilar dentro de uma faixa constante.

Mesmo os ganhos mais baixos apresentando EQMs mais atenuados, novamente é interessante destacar que a precisão desses controladores é mais alta em relação aos anteriores. A quantidade de amostras associada a cada valor de ganho do controlador PI é superior à dos controladores ON/OFF e de 3 Níveis, mas inferior a do controlador P.

Contudo, essa quantidade menor pode ser interpretada como um avanço, nesse caso. O protótipo conseguiu realizar o trajeto de uma forma mais eficiente, mesmo contando com uma quantidade de amostras levemente inferior. Desse modo, isso explica o fato do gasto energético, como também a oscilação terem sido diminuídas.

Sendo assim, o intervalo de 0,01 a 0,0001 se estabelece como a melhor opção para uma boa manutenção da dispersão, do gasto energético e da oscilação, resultando em uma precisão até mesmo acima do esperado.

4.2.4 Controlador Proporcional-Derivativo (PD):

A calibração do ganho desse controlador ocorreu de uma forma análoga a do controlador PI. Portanto, novamente o ganho proporcional foi definido em primeiro plano, e em seguida os procedimentos de ajuste para o ganho derivativo foram desenvolvidos.

Sendo assim, o ganho proporcional utilizado foi equivalente a 6, assim como no caso anterior. Com esse valor previamente definido foi necessário encontrar uma faixa de valores do ganho derivativo onde o funcionamento permitiria o comportamento adequado. Nesse caso, o controlador PD foi implementado no protótipo sem qualquer tipo de alteração em sua estrutura.

Com os testes executados a faixa de ganho aceita pelo termo derivativo estava dentro do mesmo limite encontrado para o controlador PI, isto é, compreendida entre 0 e 1. Novamente, acima de 1 o robô tinha um comportamento exagerado, e acabava executando movimentos extremamente abruptos. Logo, era impossível conduzir o seguimento do trajeto.

Optou-se por não testar valores abaixo de 0. No caso do controlador PD não existe de fato um problema caso ocorra uma mudança de sinal acompanhando os valores de erro. Sendo assim, foi mais interessante manter uma uniformidade em relação aos valores de ganho implementados.

A tabela abaixo retrata os resultados obtidos para o controlador PD em termos da dispersão:

Tabela 40 – Dispersão associada a cada valor de ganho do controlador PD.

CONTROLADOR PD				
GANHO	**MÉDIA**	**DISPERSÃO (S^2)**	**SOMA (CURVA)**	**OSILAÇÃO DO ROBÔ**
1	-4,141648271	20,90315369	40053,5	MÉDIA
0,1	-4,500740741	22,70941383	42235,95	MÉDIA
0,01	-4,161764706	18,31818379	39245,435	PEQUENA
0,001	-4,283803863	19,22831742	39299,8595	PEQUENA
0,0001	-4,245673439	20,79520905	39554,99055	PEQUENA

Fonte: Autor.

Analisando os resultados presentes na tabela acima é evidente que existe uma relação entre a diminuição do ganho derivativo, a dispersão do erro característico associado a cada valor, como também a soma e a oscilação do robô.

Conforme o ganho derivativo foi reduzido, a dispersão sofreu uma diminuição também. Essa diminuição assumiu um comportamento próprio. Novamente, não foi possível determinar a existência de uma uniformidade, pois inicialmente entre 1 e 0.1 houve um leve aumento da dispersão.

Posteriormente, há uma queda dos valores acompanhada por um novo aumento. Muito provavelmente, existe a presença de uma característica oscilatória abaixo da faixa de valor equivalente a 0.0001. Ou seja, haverá uma queda e depois um aumento, e assim sucessivamente.

Logo, a redução do ganho está condicionada a um comportamento particular da dispersão. Existem pequenos aumentos, assim como pequenas reduções. Desse modo, o início de uma característica oscilatória é evidente.

Em relação aos valores de potência (porcentagem) contabilizados na coluna curva, a mesma constatação pode ser feita, uma vez que a diminuição do ganho até a faixa de 0.01, resultou em um aumento da energia dispendida para a realização do trajeto. Contudo, uma pequena queda ocorre em seguida, e um novo aumento é observado entre 0.01 e 0.0001.

Logo, conforme o ganho é reduzido, inicialmente o gasto energético aumenta e depois sofre uma redução. A diminuição perde influência e o gasto energético passa a aumentar novamente.

Nesse controlador, o aumento e a diminuição sucessivas da dispersão é um fator bastante complexo para avaliar qual o melhor desempenho para executar o trajeto. Entretanto, é nítido também que os valores são inferiores à dos demais controladores. Outro detalhe interessante é que a quantidade de amostras, fator de determinante para análise, é inferior à do controlador P, mas próxima do controlador PI.

Na tabela abaixo, a quantidade de amostras válidas obtidas com a utilização de seu respectivo ganho é destacada:

Tabela 41 – Número de amostras associadas a cada valor de ganho do controlador PD.

CONTROLADOR PD	
GANHO	**AMOSTRAS**
1	1360
0,1	1351
0,01	1361
0,001	1347
0,0001	1330

Fonte: Autor.

Mesmo estando estabelecido um período de 10 milissegundos entre as amostras, o hardware teve uma grande dificuldade de manter esse intervalo de tempo ao longo da montagem das tabelas de dados. Consequentemente, existiu uma certa variação no intervalo de tempo.

Essa variação se mantém estável na maior parte das situações, porém repercutiu na quantidade diferente de amostras computadas para cada ganho, como no caso dos controladores anteriores.

Como a dispersão depende da quantidade de amostras para cada valor de ganho, cada vez que o ganho era reduzido, o robô tinha uma tendência de coletar mais dados, fazendo com que ele levasse mais tempo para executar o trajeto.

Contudo, nesse caso existem algumas exceções para essa premissa. O controlador PD possui a capacidade de prever o erro, e esse é um diferencial muito importante, principalmente no que diz respeito onde o robô deve realizar o seguimento da borda da linha.

Essa capacidade possui o potencial de fazer uma previsão correta, compensando o erro adequadamente, ou uma previsão incorreta, compensando o erro inadequadamente. Conforme o ganho é reduzido a compensação do erro passa a ser muito específica, ou seja, ocorre de uma maneira precisa.

Desse modo, mesmo que a compensação feita não tenha sido a melhor, ela ocasiona apenas uma pequena variação no comportamento do robô. Logo, isso influência a dispersão, como também o gasto energético, uma vez que os valores acabam presos dentro de uma faixa particular.

Ainda assim existe uma tendência do comportamento a cada nova redução, porém é mais difícil de observá-la. Sendo assim, a análise da quantidade de amostras demonstra que mesmo com um número levemente inferior ao longo dos casos, o controlador PD possui um desempenho superior.

Portanto, quanto menor o ganho, o gasto energético do robô assume um comportamento próprio, mas sua oscilação diminuí. O ganho mais baixo também influência de modo específico a capacidade do robô se locomover, e quanto mais próximo os valores estavam do limite, ou seja, do valor 1, existia um aumento do robô realizar uma compensação exagerada.

Logo, abaixo dele não existe um ponto ótimo, mas sim uma faixa de valores onde é possível fazer com que o desempenho do robô seja mais adequado a necessidade.

A tabela abaixo reforça essa ideia ao destacar os valores do erro quadrático médio característicos de cada ganho proporcional:

Tabela 42 – Erro quadrático médio associado Dispersão a cada valor de ganho do controlador PD.

CONTROLADOR PD	
GANHO	**EQM(SOMA/N)**
1	38,01305147
0,1	42,91746854
0,01	35,59882439
0,001	37,53711952
0,0001	38,77612782

Fonte: Autor.

O erro quadrático médio segue uma relação semelhante àquela observada para a dispersão, ou seja, sofre um leve aumento inicialmente, em seguida passar a ser reduzido até um certo patamar para depois voltar a ser elevado.

Mesmo os ganhos mais baixos apresentando EQMs seguindo um comportamento próprio, novamente é interessante destacar que a precisão desses controladores é mais alta em relação aos anteriores.

A quantidade de amostras associada a cada valor de ganho do controlador PD é superior à dos controladores ON/OFF e de 3 Níveis, mas inferior à do controlador P e semelhante a do controlador PI.

Essa quantidade menor pode ser interpretada como um avanço, nesse caso. O protótipo conseguiu realizar o trajeto de uma forma mais eficiente, mesmo contando com uma quantidade de amostras levemente inferior.

4.2.5 Controlador Proporcional-Integral-Derivativo (PID):

A calibração do ganho do controlador PID foi baseada no processo empregado para os controladores PI e PD, ou seja, primeiro calibrou-se o ganho proporcional, e depois os ganhos integral e derivativo.

O ganho proporcional utilizado permaneceu equivalente a 6. Já os ganhos integral e derivativo assumiram a faixa de valores entre 1 e 0.0001. Com esses valores previamente definidos uma série de testes foram executados no protótipo buscando encontrar a melhor faixa de combinação.

Ao longo dos testes, ficou evidente que as capacidades do controlador PID são superiores em relação aos demais. Mais precisamente, o controlador possibilitou ao protótipo executar o trajeto de uma maneira extremamente precisa (acurada).

O robô sempre se manteve preso a borda da fita isolante preta, tanto nos trechos retilíneos, quanto nos trechos curvilíneos. O erro característico é melhor compensado a cada nova amostra, uma vez que é possível avaliar o erro atual, comparando com os erros das amostras passadas e prevendo o erro das amostras futuras.

A tabela abaixo retrata os resultados obtidos para o controlador PID em termos da dispersão:

Tabela 43 – Dispersão associada a diferentes valores de ganhos P e I fixos e a ganho D variável.

CONTROLADOR PID						
GANHO P	**GANHO I**	**GANHO D**	**MÉDIA**	**DISPERSÃO(S^2)**	**SOMA(CURVA)**	**OSILAÇÃO DO ROBÔ**
6	**1**	**1**	3,46847912	13,52672848	39450,26945	PEQUENA
6	**1**	**0,1**	3,49295775	13,6048259	37812,04815	PEQUENA
6	**1**	**0,01**	3,43183579	13,31361753	39031,59462	PEQUENA
6	**1**	**0,001**	3,58027523	14,32154489	36851,95267	PEQUENA
6	**1**	**0,0001**	3,52684049	13,89818917	39938,97737	PEQUENA

GANHO P	**GANHO I**	**GANHO D**	**MÉDIA**	**DISPERSÃO(S^2)**	**SOMA(CURVA)**	**OSILAÇÃO DO ROBÔ**
6	**0,1**	**1**	3,73478261	15,87823335	39360,87888	PEQUENA
6	**0,1**	**0,1**	3,77675841	15,91490832	40416,55708	PEQUENA
6	**0,1**	**0,01**	3,77447957	15,91951809	41233,99573	PEQUENA
6	**0,1**	**0,001**	3,64918415	14,92317571	42042,78645	PEQUENA
6	**0,1**	**0,0001**	3,72804314	15,50715976	41067,79081	PEQUENA

GANHO P	**GANHO I**	**GANHO D**	**MÉDIA**	**DISPERSÃO(S^2)**	**SOMA(CURVA)**	**OSILAÇÃO DO ROBÔ**
6	**0,01**	**1**	3,95590551	17,55743161	42783,23062	PEQUENA
6	**0,01**	**0,1**	3,8351227	16,28383885	42142,7985	PEQUENA
6	**0,01**	**0,01**	3,74825446	15,7342528	40441,58668	PEQUENA
6	**0,01**	**0,001**	3,69583655	15,32576899	41227,10183	PEQUENA
6	**0,01**	**0,0001**	3,76733436	15,74058572	39907,33872	PEQUENA

GANHO P	GANHO I	GANHO D	MÉDIA	DISPERSÃO(S^2)	SOMA(CURVA)	OSILAÇÃO DO ROBÔ
6	**0,001**	**1**	3,79844961	16,31657405	43147,29786	PEQUENA
6	**0,001**	**0,1**	3,69770992	15,1091571	40045,49577	PEQUENA
6	**0,001**	**0,01**	3,76332288	16,0073176	41792,76303	PEQUENA
6	**0,001**	**0,001**	3,72583082	15,41034463	42543,82803	PEQUENA
6	**0,001**	**0,0001**	3,87575758	16,37489719	41206,3294	PEQUENA

GANHO P	GANHO I	GANHO D	MÉDIA	DISPERSÃO(S^2)	SOMA(CURVA)	OSILAÇÃO DO ROBÔ
6	**0,0001**	**1**	3,54914197	14,22271413	41230,27592	PEQUENA
6	**0,0001**	**0,1**	3,93138801	17,19019256	42610,71276	PEQUENA
6	**0,0001**	**0,01**	3,87575758	16,37489719	41206,3294	PEQUENA
6	**0,0001**	**0,001**	3,85932722	16,46460894	41963,68832	PEQUENA
6	**0,0001**	**0,0001**	3,94314642	18,02825652	45431,74819	PEQUENA

Fonte: Autor.

Analisando os resultados presentes no conjunto de tabelas acima, onde o ganho proporcional é constante e o ganho integrativo também, é possível observar que no controlador PID a faixa de valores da dispersão, assim como do gasto energético são constantes.

Existe a presença de uma leve variação conforme o ganho derivativo foi reduzido, contudo ela não é de fato muito significativa. Os valores já estão em uma faixa abaixo dos resultados encontrados no caso dos controladores anteriores, e a influência sobre a oscilação do robô já não é mais percebida.

Basicamente, essa modalidade de controlador é a que deveria apresentar o comportamento mais uniforme, coeso e coerente em teoria. Os dados acima reiteram essa ideia, pois a verificação de alguma tendência de comportamento é mais difícil de ser evidenciada.

O mesmo pode ser verificado na situação em que o ganho proporcional foi mantido constante e o ganho derivativo também, enquanto apenas o ganho integral era variado. As tabelas a seguir destacam essa outra bateria de testes:

Tabela 44 – Dispersão associada a diferentes valores de ganhos P e D fixos e a ganho I variável.

CONTROLADOR PID						
GANHO P	**GANHO I**	**GANHO D**	**MÉDIA**	**DISPERSÃO(S^2)**	**SOMA(CURVA)**	**OSILAÇÃO DO ROBÔ**
6	**1**	**1**	3,46847912	13,52672848	39450,26945	PEQUENA
6	**0,1**	**1**	3,73478261	15,87823335	39360,87888	PEQUENA
6	**0,01**	**1**	3,95590551	17,55743161	42783,23062	PEQUENA
6	**0,001**	**1**	3,79844961	16,31657405	43147,29786	PEQUENA
6	**0,0001**	**1**	3,54914197	14,22271413	41230,27592	PEQUENA

GANHO P	**GANHO I**	**GANHO D**	**MÉDIA**	**DISPERSÃO(S^2)**	**SOMA(CURVA)**	**OSILAÇÃO DO ROBÔ**
6	**1**	**0,1**	3,49295775	13,6048259	37812,04815	PEQUENA
6	**0,1**	**0,1**	3,77675841	15,91490832	40416,55708	PEQUENA
6	**0,01**	**0,1**	3,8351227	16,28383885	42142,7985	PEQUENA
6	**0,001**	**0,1**	3,69770992	15,1091571	40045,49577	PEQUENA
6	**0,0001**	**0,1**	3,93138801	17,19019256	42610,71276	PEQUENA

GANHO P	**GANHO I**	**GANHO D**	**MÉDIA**	**DISPERSÃO(S^2)**	**SOMA(CURVA)**	**OSILAÇÃO DO ROBÔ**
6	**1**	**0,01**	3,43183579	13,31361753	39031,59462	PEQUENA
6	**0,1**	**0,01**	3,77447957	15,91951809	41233,99573	PEQUENA
6	**0,01**	**0,01**	3,74825446	15,7342528	40441,58668	PEQUENA
6	**0,001**	**0,01**	3,76332288	16,0073176	41792,76303	PEQUENA
6	**0,0001**	**0,01**	3,87575758	16,37489719	41206,3294	PEQUENA

GANHO P	**GANHO I**	**GANHO D**	**MÉDIA**	**DISPERSÃO(S^2)**	**SOMA(CURVA)**	**OSILAÇÃO DO ROBÔ**
6	**1**	**0,001**	3,58027523	14,32154489	36851,95267	PEQUENA
6	**0,1**	**0,001**	3,64918415	14,92317571	42042,78645	PEQUENA
6	**0,01**	**0,001**	3,74825446	15,7342528	40441,58668	PEQUENA
6	**0,001**	**0,001**	3,72583082	15,41034463	42543,82803	PEQUENA
6	**0,0001**	**0,001**	3,94314642	18,02825652	45431,74819	PEQUENA

GANHO P	**GANHO I**	**GANHO D**	**MÉDIA**	**DISPERSÃO(S^2)**	**SOMA(CURVA)**	**OSILAÇÃO DO ROBÔ**
6	**1**	**0,0001**	3,52684049	13,89818917	39938,97737	PEQUENA
6	**0,1**	**0,0001**	3,72804314	15,50715976	41067,79081	PEQUENA
6	**0,01**	**0,0001**	3,76733436	15,74058572	39907,33872	PEQUENA
6	**0,001**	**0,0001**	3,87575758	16,37489719	41206,3294	PEQUENA
6	**0,0001**	**0,0001**	3,94314642	18,02825652	45431,74819	PEQUENA

Fonte: Autor.

Novamente, é nítido que as relações entre o comportamento da dispersão e do gasto energético já não se aplicam como nos casos anteriores. A variação existente entre os valores é pequena e não chega a repercutir diretamente na oscilação do robô, por exemplo.

Logo, a redução do ganho não está condicionada a um comportamento particular da dispersão, ou até mesmo do gasto energético. Nesse controlador, o aumento e a diminuição sucessivas da dispersão é um fator bastante suave.

A única atenção necessária é em relação aos ganhos superiores a 1. Acima desse limite grandes problemas acontecem com a capacidade de compensação do erro, porém eles não são diferentes daqueles já salientados para os controladores anteriores.

A quantidade de amostras correspondente a cada conjunto de valores implementados no ganho também não marca uma presença muito expressiva. Na tabela abaixo, a quantidade de amostras válidas obtidas com a utilização de seu respectivo ganho é destacada:

Tabela 45 – Quantidade de amostras associadas a diferentes valores de ganhos P e I fixos e a ganho D variável.

CONTROLADOR PID			
GANHO P	**GANHO I**	**GANHO D**	**AMOSTRAS**
6	**1**	**1**	1270
6	**1**	**0,1**	1279
6	**1**	**0,01**	1292
6	**1**	**0,001**	1309
6	**1**	**0,0001**	1305

GANHO P	**GANHO I**	**GANHO D**	**AMOSTRAS**
6	**0,1**	**1**	1266
6	**0,1**	**0,1**	1309
6	**0,1**	**0,01**	1298
6	**0,1**	**0,001**	1288
6	**0,1**	**0,0001**	1299

GANHO P	**GANHO I**	**GANHO D**	**AMOSTRAS**
6	**0,01**	**1**	1271
6	**0,01**	**0,1**	1305
6	**0,01**	**0,01**	1290
6	**0,01**	**0,001**	1298
6	**0,01**	**0,0001**	1299

GANHO P	GANHO I	GANHO D	AMOSTRAS
6	**0,001**	**1**	1291
6	**0,001**	**0,1**	1311
6	**0,001**	**0,01**	1277
6	**0,001**	**0,001**	1325
6	**0,001**	**0,0001**	1321

GANHO P	GANHO I	GANHO D	AMOSTRAS
6	**0,0001**	**1**	1283
6	**0,0001**	**0,1**	1278
6	**0,0001**	**0,01**	1269
6	**0,0001**	**0,001**	1309
6	**0,0001**	**0,0001**	1285

Fonte: Autor.

Tabela 46 - Quantidade de amostras associadas a diferentes valores de ganhos P e D fixos e a ganho I variável.

CONTROLADOR PID			
GANHO P	**GANHO I**	**GANHO D**	**AMOSTRAS**
6	**1**	**1**	1270
6	**0,1**	**1**	1266
6	**0,01**	**1**	1271
6	**0,001**	**1**	1291
6	**0,0001**	**1**	1283

GANHO P	GANHO I	GANHO D	AMOSTRAS
6	**1**	**0,1**	1279
6	**0,1**	**0,1**	1309
6	**0,01**	**0,1**	1305
6	**0,001**	**0,1**	1311
6	**0,0001**	**0,1**	1278

GANHO P	GANHO I	GANHO D	AMOSTRAS
6	**1**	**0,01**	1292
6	**0,1**	**0,01**	1298
6	**0,01**	**0,01**	1290
6	**0,001**	**0,01**	1277
6	**0,0001**	**0,01**	1269

GANHO P	GANHO I	GANHO D	AMOSTRAS
6	**1**	**0,001**	1309
6	**0,1**	**0,001**	1288
6	**0,01**	**0,001**	1298
6	**0,001**	**0,001**	1325
6	**0,0001**	**0,001**	1309

GANHO P	GANHO I	GANHO D	AMOSTRAS
6	**1**	**0,0001**	1305
6	**0,1**	**0,0001**	1299
6	**0,01**	**0,0001**	1299
6	**0,001**	**0,0001**	1321
6	**0,0001**	**0,0001**	1285

Fonte: Autor.

Mesmo estando estabelecido um período de 10 milissegundos entre as amostras, o hardware teve uma grande dificuldade de manter esse intervalo de tempo ao longo da montagem das tabelas de dados. Consequentemente, existiu uma certa variação no intervalo de tempo.

Essa variação se mantém estável na maior parte das situações, porém repercutiu na quantidade de amostras computadas para cada ganho, como no caso dos controladores anteriores.

Como a dispersão depende da quantidade diferente de amostras para cada valor de ganho, cada vez que o ganho era reduzido, o robô tinha uma tendência de coletar mais dados, fazendo com que ele levasse mais tempo para executar o trajeto.

Nesse caso, esse fenômeno também é difícil de enxergar. O controlador PID possui a capacidade de avaliar o erro atual, com base no erro passado e até mesmo prever o erro futuro. Portanto, ele é a modalidade mais completa, e esse acaba sendo um diferencial muito importante com relação ao que robô deve realizar durante o seguimento da borda da linha.

Essas três capacidades somadas permitem que o potencial de fazer o seguimento de linha, compensando o erro adequadamente a cada nova leitura possua uma precisão acima dos demais. O principal elemento que destaca essa característica é a suavidade com que ele consegue se deslocar nos trechos retilíneos e nos trechos curvilíneos do trajeto.

Desse modo, a compensação acaba sendo a melhor possível, uma vez que ela avalia três possíveis vertentes que impactam no erro da leitura atual e consequentemente na atuação dos motores a ser conduzida.

Logo, como não existe necessidade de grandes valores de atuação isso influência a dispersão, como também o gasto energético, uma vez que os valores acabam presos dentro de uma faixa particular.

As amostras marcam uma maior influência sobre os EQMs. As tabelas abaixo reforçam essa ideia ao destacar os valores do erro quadrático médio característicos de cada combinação de ganho, com o ganho proporcional e o ganho integral constantes, e o ganho derivativo variando:

Tabela 47 - Erro quadrático associado a diferentes valores de ganhos P e I fixos e a ganho D variável.

CONTROLADOR PID			
GANHO P	**GANHO I**	**GANHO D**	**EQM(SOMA/N)**
6	**1**	**1**	52,41358268
6	**1**	**0,1**	45,83698202
6	**1**	**0,01**	47,17859907
6	**1**	**0,001**	42,89915966
6	**1**	**0,0001**	47,41609195

GANHO P	**GANHO I**	**GANHO D**	**EQM(SOMA/N)**
6	**0,1**	**1**	47,04601106
6	**0,1**	**0,1**	45,94576012
6	**0,1**	**0,01**	49,06953005
6	**0,1**	**0,001**	49,08831522
6	**0,1**	**0,0001**	46,19899923

GANHO P	**GANHO I**	**GANHO D**	**EQM(SOMA/N)**
6	**0,01**	**1**	54,05782848
6	**0,01**	**0,1**	51,4559387
6	**0,01**	**0,01**	46,90096899
6	**0,01**	**0,001**	49,24210324
6	**0,01**	**0,0001**	43,54157044

GANHO P	**GANHO I**	**GANHO D**	**EQM(SOMA/N)**
6	**0,001**	**1**	53,11890008
6	**0,001**	**0,1**	44,78070175
6	**0,001**	**0,01**	51,21769773
6	**0,001**	**0,001**	48,64377358
6	**0,001**	**0,0001**	44,66161998

GANHO P	GANHO I	GANHO D	EQM(SOMA/N)
6	**0,0001**	**1**	48,82346064
6	**0,0001**	**0,1**	46,19899923
6	**0,0001**	**0,01**	50,05122143
6	**0,0001**	**0,001**	47,93659282
6	**0,0001**	**0,0001**	57,16964981

Fonte: Autor.

Já esse outro conjunto de tabelas demonstra essa mesma ideia, contudo com cada combinação de ganho proporcional e derivativo constantes, e o ganho integral variando:

Tabela 48 - Erro quadrático médio associado a diferentes valores de ganhos P e D fixos e a ganho I variável.

CONTROLADOR PID			
GANHO P	**GANHO I**	**GANHO D**	**EQM(SOMA/N)**
6	**1**	**1**	52,41358268
6	**0,1**	**1**	47,04601106
6	**0,01**	**1**	54,05782848
6	**0,001**	**1**	53,11890008
6	**0,0001**	**1**	48,82346064

GANHO P	GANHO I	GANHO D	EQM(SOMA/N)
6	**1**	**0,1**	45,83698202
6	**0,1**	**0,1**	45,94576012
6	**0,01**	**0,1**	51,4559387
6	**0,001**	**0,1**	44,78070175
6	**0,0001**	**0,1**	46,19899923

GANHO P	GANHO I	GANHO D	EQM(SOMA/N)
6	**1**	**0,01**	47,17859907
6	**0,1**	**0,01**	49,06953005
6	**0,01**	**0,01**	46,90096899
6	**0,001**	**0,01**	51,21769773
6	**0,0001**	**0,01**	50,05122143

GANHO P	GANHO I	GANHO D	EQM(SOMA/N)
6	**1**	**0,001**	42,89915966
6	**0,1**	**0,001**	49,08831522
6	**0,01**	**0,001**	49,24210324
6	**0,001**	**0,001**	48,64377358
6	**0,0001**	**0,001**	47,93659282

GANHO P	GANHO I	GANHO D	EQM(SOMA/N)
6	**1**	**0,0001**	47,41609195
6	**0,1**	**0,0001**	46,19899923
6	**0,01**	**0,0001**	43,54157044
6	**0,001**	**0,0001**	44,66161998
6	**0,0001**	**0,0001**	57,16964981

Fonte: Autor.

O erro quadrático médio ficou mais suscetível a uma flutuação de seus respectivos valores devido a quantidade de amostras, como também ao fato de o erro ser avaliado em três frentes.

Mesmo os valores tendo ficado mais elevados, e superando em alguns casos o controlador P, PI e PD é necessário levar em consideração também o melhor desempenho do robô ao longo do trajeto.

O grande avanço desse controlador está diretamente ligado ao fato de o robô realizar movimentos mais suaves devido ao modo como o erro é interpretado. O protótipo conseguiu realizar o trajeto e se ater a referência muito mais facilmente se comparado aos outros casos.

5 CONCLUSÃO:

Esta seção do trabalho buscar evidenciar os principais pontos alcançados ao longo do seu respectivo processo de desenvolvimento. A criação de um robô seguidor de linha e a implementação de diferentes tipos controladores, com o objetivo de avaliar cada um dos desempenhos, é um estudo bastante relevante para área de automação, controle e robótica. Contudo, extremamente complexo, quando desenvolvido com profundidade.

Mais precisamente, a construção de um robô seguidor de linha e a elaboração de sua programação não são tarefas difíceis, principalmente quando é levado em consideração o fato desse processo se consolidar como o primeiro contato de muitos estudantes de nível básico, e de nível superior, com essa área de conhecimento.

A grande questão é que em muitos casos, busca-se simplesmente fazer com que o robô siga um dado circuito, sem se preocupar com questões mais implícitas relacionadas ao seu desempenho. Com a disseminação dos microcontroladores embarcados, dos kits de robótica, como, por exemplo, o LEGO ® MINDSTORMS Ev3, houve uma contribuição, para que esse desafio fosse encarado de modo relativamente mais acessível.

Sendo assim, esse trabalho pode trazer à tona nuances intricadas que muitas vezes são negligenciadas. Mais precisamente, ainda existe uma grande dificuldade de fazer com que os robôs tenham a capacidade de perceber os diferentes aspectos presentes no mundo ao seu redor.

Atualmente, mesmo existindo uma variedade de tecnologias de ponta, de diferentes custos, ainda é uma tarefa muito difícil fazer com que um determinado protótipo tenha autonomia em sua respectiva tomada de decisão.

O simples de fato de o robô seguidor de linha ter de se movimentar à direita ou à esquerda da borda utilizada com referência é um grande desafio, pois envolve a criação de uma lógica de funcionamento que abrange desde o modelamento do sensor de luz, até o fornecimento de tensão para cada um dos atuadores empregados na montagem.

Portanto, existe uma vasta necessidade de aprofundar os conhecimentos relacionados a essa interação. A modelagem do comportamento dos motores de corrente contínua considerando seus aspectos mecânicos, assim como seus aspectos elétricos, revelou-se uma tarefa que exigiu uma compreensão profunda das relações dinâmicas envolvidas. Principalmente, no que diz respeito a equiparação entre o comportamento avaliado no mundo real e no mundo virtual.

Conforme, destacado ao longo das seções foi necessário buscar softwares de alto desempenho para realizar essa equiparação. Mesmo existindo uma grande variedade, foi possível perceber que eles apresentavam limitações, exigindo o uso de mais de um, para que fosse possível obter resultados consistentes.

Um grande exemplo disso foi a necessidade de fazer o Matlab e o Visual Studio Code trabalharem em conjunto para realizar os procedimentos de verificação das características de cada um parâmetro dos motores, como também a construção dos algoritmos associados a cada um dos controladores.

A integração desses elementos, muitas vezes subestimada, é o cerne da verdadeira eficácia de um robô seguidor de linha. Dessa forma, a todo momento ficou bastante evidente a demanda de uma abordagem meticulosa para capturar com precisão a complexidade das informações do sistema, do ambiente ao se redor, assim como a maneira pela qual esses elementos interagiam.

Além disso, outro aspecto importante foi a percepção da existência de diferenças inerentes a cada um desses ambientes, ou seja, mesmo com a instalação de pacotes de conteúdo nos dois softwares, ainda sim existiram diferenças no comportamento ideal avaliado na simulação virtual, em relação ao comportamento real.

O Matlab com pacotes do SIMULINK, “Mobile Robotics Training” e “LEGO MINDSTORMS EV3 Hardware”, e a simulação no protótipo real utilizando o Visual Studio Code com a extensão “Micropython” e “ev3dev”, foram determinantes para destacar essas diferenças.

A discrepância, em questão, sublinha a importância de considerar os desafios do mundo real, onde variáveis imprevistas e condições não simuladas podem afetar significativamente o desempenho do robô. Somado a esse fenômeno também existe a análise dos controladores.

No contexto do robô seguidor de linha, a implementação de cada um dos algoritmos revelou nuances distintas entre os sistemas ON/OFF, de 3 Níveis e os controladores P, PI, PD e PID.

Os controladores ON/OFF e de 3 Níveis, de abordagem mais simples, demonstraram um desempenho inferior quando comparados à complexidade e adaptabilidade dos controladores PID. Entretanto, ainda assim conseguiriam executar a tarefe central, ou seja, realizar o seguimento do trajeto.

A estratégia ON/OFF e de 3 Níveis, embora eficazes em situações binárias, revelaram-se limitadas na capacidade de proporcionar um controle suave e preciso. Em contraste, os controladores P, PI, PD e PID oferecem uma gama mais ampla de ajustes e respostas, permitindo uma adaptação mais refinada às condições variáveis encontradas em um ambiente real.

Ao analisar especificamente os controladores P, PI, PD e PID, é evidente que o PID emerge como o líder indiscutível em termos de desempenho. Sua capacidade de equilibrar o ganho proporcional, integral e derivativo desencadeia um controle robusto e eficiente.

Os controladores PD e PI ocupam o segundo lugar, com suas respectivas ênfases na compensação de erros por meio do ganho derivativo ou do ganho integral, demonstrando eficácia em diferentes contextos.

Surpreendentemente, o controlador P, que se baseia apenas na proporção, fica em terceiro lugar. Isso destaca a importância das contribuições integral e derivativa na melhoria do desempenho, fornecendo um dado valioso para futuras otimizações e ajustes no sistema.

Assim, a conclusão não apenas confirma a superioridade dos controladores P, PI, PD e PID sobre os ON/OFF e de 3 Níveis, mas também oferece uma hierarquia interna entre esses controladores avançados.

Esta diferenciação proporciona um guia valioso para a implementação prática de estratégias de controle, enfatizando a necessidade de considerar a complexidade das interações envolvidas no processo de seguir linha de forma eficaz e adaptativa.

Em última análise, a conclusão deste trabalho destaca que a construção e o estudo de controladores em um robô seguidor de linha transcendem a simples montagem de peças de LEGO.

Ela requer um entendimento profundo e integrado das disciplinas de elétrica, de mecânica, de programação e de processamento de dados, de automação, de controle, como também de robótica.

A busca por eficiência não pode prescindir da consideração minuciosa das complexas interações entre hardware, software e ambiente. Este estudo oferece não apenas uma contribuição para o campo da robótica educacional, mas também ressalta a importância de abordar desafios aparentemente simples com a devida profundidade e meticulosidade.

5.1 Sugestões para futuros trabalhos:

Considerando os aspectos relacionados a profundidade e a complexidade do tema abordado nesse trabalho, existe um conjunto de possibilidades voltadas para interessantes para a condução de desenvolvimentos, estudos, além de pesquisas futuras. e desenvolvimentos.

Em primeiro plano é importante destacar que alguns elementos presentes na estrutura básica do protótipo, como, por exemplo o sensor infravermelho, não foram empregados ao longo do estudo conduzido para o robô seguidor de linha. Dessa forma, o modelamento do comportamento desse sensor e aplicação em conjunto com o ideal do robô seguidor de linha é uma ótima área de estudo, uma vez que viabilizaria a capacidade de desvio de obstáculos, valorizando o ideal associado a um comportamento cada vez mais autônomo.

Seguindo essa mesma lógica implementar algoritmos voltados para o emprego de inteligência artificial, assim como aprendizado de máquina, também constituem uma ótima possibilidade de estudo, pois garantiram a adaptação dinâmica a do robô a diferentes condições de pista e ambientes.

Paralelamente, também seria interessante conduzir pesquisas relacionadas com uma equiparação entre o comportamento virtual e o comportamento real avaliados, ou seja, o que seria possível fazer para que a margem de ajuste de ganho nas duas situações fosse equivalente, apresentando uma discrepância mais limitada.

Sendo assim, a visão computacional poderia auxiliar nesse processo. Com uma melhor detecção, mas também resposta aos padrões de trajetos mais complexos, o comportamento característico robô estaria cada vez mais próximo do ideal. Logo, essa frente também possibilitaria um melhor aproveitamento do sensor infravermelho no reconhecimento de obstáculos, ou de características específicas do ambiente.

Outra possibilidade de pesquisa seria o estabelecimento de uma comunicação entre robôs seguidores de linha. Por exemplo, considerando uma aplicação onde muitos robôs estejam se comunicando, a investigação de estratégias, por exemplo para coordenar vários grupos, é uma alternativa versátil para a detecção de padrão particulares, possibilitando resolução de problemas, abrangendo a aplicação da robótica colaborativa.

Essa aplicação fundamenta outro campo não explorado em total profundidade, isto é, a interação entre seres humanos e robôs. Atualmente, essa interação ainda ocorre de um modo simples, portanto ainda é possível aumentar a eficácia e a viabilidade com o trabalho exercido pela mão de obra humana. Assim, além de seguir linha, o robô poderia executar outras tarefas em conjunto com uma pessoa.

Outra área interessante diz respeito aos ambientes, ou espaços em que trajeto se encontra. Esses trajetos se materializam na forma de Ambientes Não-Estruturados, ou seja, expandir as funcionalidades do robô para situações fora do padrão. Desse modo, simular em diferentes tipos de superfícies, como também realizar trajetos com curvas mais complexas, ou condições de iluminação adversas poderiam consolidar casos importantes de desenvolvimento.

Por último, o uso de outros kits de robótico e de outros softwares dedicados poderia ser uma ótima forma de avaliar a aplicabilidade do sistema construído procurando considerar outras possibilidades de resolução.

Logo, com um novo material a prototipagem poderia ser fundamental para conduzir testes de desempenho em ambientes dinâmicos com obstáculos móveis, representando situações do mundo real. Ao mesmo tempo, o uso de um novo software poderia facilitar o projeto de uma interface de usuário, para permitir que uma maior quantidade de pessoas pudesse interagir com a tecnologia e aplicá-la conforme suas necessidades, configurando e personalizando facilmente o comportamento do robô seguidor de linha.

6 BIBLIOGRAFIA:

1. THE LEGO GROUP. LEGO® Education WeDo 2.0. **LEGO® Education WeDo 2.0**, 2018. Disponivel em: <https://le-www-live-s.legocdn.com/sc/media/files/user-guides/wedo-2/teacher-guides/teacherguide-pt-br-v1-0f551d9951d95fa52bb963871915e70b.pdf>. Acesso em: 08 maio 2023.

2. VAZ, M. A. Lego Education Mindstorms Ev3 – 45544. **Nitmaker**. Disponivel em: <https://nitmaker.com.br/produtos/lego-education-mindstorms-ev3-45544/>. Acesso em: 23 ago. 2023.

3. TANAKA, A. H. S.; SCHERWINSKI, A. T.; MELO, M. J. Controlador PID Aplicado ao Rastreamento de Trajetória. **Mostra Nacional de Robótica (MNR)**, 2019.

4. SIEGWART, R.; NOURBAKHSH, I. R. **Introduction to Autonomous Mobile Robots**. Cambrige, Massachusetts - London, England: MIT Press (MA), 2004.

5. SCHWAB, K. **The Fourth Industrial Revolution**. Cologny/Geneva: World Economic Forum, 2016.

6. RIBEIRO, T. J. B. Controlador PID Aplicado a Robótica Móvel. **Mostra Nacional de Robótica (MNR0**, 2016.

7. RIBEIRO, T. J. B.; KASHIWAGI, M. Controlador PID Aplicado a Programação de Robô Móvel. **Congresso de Inovação, Ciência e Tecnologia do IFSP**, 2016.

8. RIBEIRO, M. A. **Automação Industrial**. 4ª Edição. ed. Salvador: Tek Treinamento e Consultoria Ltda, 2001.

9. OGATA, K. **Egenharia de Controle Moderno**. 5ª Edição. ed. São Paulo: Pearson Education do Brasi Ltda, 2014.

10 NISE, N. S. **Engenharia de Sistemas de Controle**. 6ª Edição. ed. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora Ltda. Uma editora integrante do GEN | Grupo Editorial Nacional, 2013.

11 NIKU, S. B. **Introduction to Robotics, Analysis, Systems, Applications**. [S.l.]: Prentice Hall Inc, 2001.

12 NELSON, W.; COX, I. Local Path Control for an Autonomous Vehicle. **Robotics and Automation, IEEE International Conference**, v. 3, p. 1504-1510, 1988.

13 NEHMZOW, U. **Mobile Robotics:** A Pratical Introduction. [S.l.]: Springer, 2002.

14 MUIR, P.; NEWMAN, C. Kinematic Modeling for Feedback Control of an Omnidirectional Wheeled Mobile Robot. **Robotics and Automation, IEEE International Conference**, v. 4, p. 1772-1778, 1987.

15 MONTENEGRO, A. B. et al. Controlador PID em Robô Seguidor de Linha em Plataforma LEGO EV3.

16 MIRANDA, A. R. L. **Comparação entre Controladores Clássicos e Fuzzy Comandando um Pêndulo Invertido Tipo Carro ImplemEntado com LEGO MINDSTORM EV3**. Universidade Federal de Ouro Preto - Escola de Minas Gerais Colegiado do Curso de Engenharia de Controle e Automação - CECAU. Ouro Preto. 2015.

17 LOZANO-PÉREZ, T. **Autonomous Robot Vehicles**. [S.l.]: Springer-Verlag, 1990.

18 KERAMAS, J. G. **Robot Technology Fundamentals**. [S.l.]: Cenage Learning, 1998.

19 KANAYAMA, Y.; HARTMAN, B. Smooth Local Path Planning for Autonomous Vehicles. **International Journal of Robotics Research**, v. 16, p. 263-284, 1987.

20 JÚNIOR, V. F. **Desenvolvimento e Cooperação de Robôs Através da Plataforma Arduino**. Universidade Federal de Santa Catarina - Centro Tecnológico Departamento de Automação e Sistemas. Florianópolis. 2018.

21 J, S. A PID Controller For Lego Mindstorms Robots. **Inpharmix**, 2018. Disponivel em: <https://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html> . Acesso em: 16 jun. 2023.

22 HONGO, T. et al. An Automatic Guidance System of a Self-Controlled Vehicle. **Industrial Eletronics, IEEE Transactions**, v. 44, p. 5-10, 1987.

23 GROUP, L. LEGO MINDSTORMS EV3 Hardware Developer Kit. **Mikcrontroller**, 2013. Disponivel em: <https://www.mikrocontroller.net/attachment/338591/hardware_developer_kit.pdf>. Acesso em: 20 jul. 2023.

24 GROUP, L. LEGO® MINDSTORMS® firmware Developer Kit. **lego.com**, 2013. Disponivel em: <https://education.lego.com/v3/assets/blt293eea581807678a/bltb470b9ea6e38f8d4/5f8802fc4376310c19e33714/getting-started-with-micropython-v2_enus.pdf>. Acesso em: 18 jul. 2023.

25 GROUP, L. LEGO® MINDSTORMS® Education EV3. **LEGO® Education**. Disponivel em: <https://education.lego.com/pt-br/product-resources/mindstorms-ev3/downloads/developer-kits>. Acesso em: 18 jul. 2023.

26 GROUP, L. Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython. **LEGO® Education**, 2019-2020. Disponivel em: <8f8d4/5f8802fc4376310c19e33714/getting-started-with-micropython-v2_enus.pdf>. Acesso em: 18 jul. 2023.

27 GROUP, L. Building instructions for 31313, LEGO® MINDSTORMS® EV3. **lego.com**, 2013. Disponivel em: <https://www.lego.com/en-us/service/buildinginstructions/31313>. Acesso em: 17 jul. 2023.

28 GOVINDARAJAN, M.; MATHWORKS. Obstacle Avoidance using LEGO Mindstorms EV3 and Simulink. **MathWorks**, 2016. Disponivel em: <https://www.mathworks.com/matlabcentral/fileexchange/56293-obstacle-avoidance-using-lego-mindstorms-ev3-and-simulink?s_tid=mwa_osa_a>. Acesso em: 24 jul. 2023.

29 GOLNARIGHI, F.; KUO, B. C. **Automatic Control Systems**. 9ª Edição. ed. [S.l.]: John Wiley & Sons, Inc., 2009.

30 FRANKLIN, G. F.; POWELL, J. D.; EMAMI-NAEINI, A. **Feedback Control of Dynamic Systems**. 6ª Edição. ed. [S.l.]: Pearson - Prentice Hall.

31 EVANS, G. W. Bringing root locus to the classroom. **IEEE Xplorre**, 2004. Disponivel em: <https://ieeexplore.ieee.org/document/1368483>. Acesso em: 27 abr. 2023.

32 CRAIG, J. J. **Introduction to Robotics:** Mechanics and Control. 3ª Edição. ed. [S.l.]: Pearson Educational Inc., 2005.

33 CARVALHO, F. C. et al. Projeto de Construção e Controle PID de um Manipulador Robótico para Auxílio na Aprendizagem de Alunos de Graduação em Engenharia. **Revista SBA: Controle e Automação.**

34 BRYNJOLFSSON, E.; MACFEE, A. **The Second Machine Age:** Work, Progress and Prosperity in a Time of Brillant Technologies. 1ª Edição. ed. New York and London: W.W.Norton & Company, 2014.

35 BRITO, R. C.; MADALOSSO, E.; GUIBES, G. A. O. Seguidor de Linha Para LEGO ® Mindstorms Utilizando Controle PID. **Computer on the Beach**, p. 310-319, 2014.

36 BOLTON, W. **Engenharia de Controle**. São Paulo: Makron Books do Brasil Editora Ltda, 1995.

37 BLADIN, P. F. W. Grey Walter, pioneer in the electroencephalogram, robotics, cybernetics, artificial intelligence. **Science Direct**, 2005. Disponivel em: <https://www.sciencedirect.com/science/article/abs/pii/S096758680500398X>. Acesso em: 27 abr. 2023.

38 BENNET, S. **A History Of Control Engineering 1930 - 1955**. London: Peter Peregrinus Ltd. on behalf of the Instituion of Eletrical Engineers, 1993.

39 BARBOSA, A. C. **Robô LEGO NXT Seguidor de Trajetória com Controle Fuzzy**. Universidade Estadual de Londrina - Centro de Tecnologia e Urbanismo Departamento de Engenharia Elétrica. Londrina. 2017.

40 BANDEIRA, M. S.; CARNEIRO, R. D. S. **Projeto Básico de Robô Seguidor de Linha Controlado por Arduino**. Universidade Federal Fluminense - Escola de Engenharia. Niterói. 2021.

41 ASIMOV, I. **Eu Robô**. 2ª Edição. ed. [S.l.]: Aliança OCR Brasil, 1969.

42 ALEXANDER, J. C.; MADDOCKS, J. H. On Kinematics of Wheeled Mobile Robots. **International Journal of Robotic Research - IJRR**, v. 8, p. 15-27, 1989.

43 AHARARI, A.; UEDA, Y. **Low Pass Filter Applied to Color Sensor of Line Followe Robot**. Procedia Computer Science. [S.l.]: [s.n.]. 2018. p. 693-698.

44 ÂNSTRÖM, K. J.; KUMAR, P. R. Control: A perspective. **Automatica**, v. 50, n. 1, p. 3-43, 2014. ISSN ISSN: 00051098.

45 (陈新宇), X. C. Understanding Lyapunov Equation through Kronecker Product and Linear Equation. **Towards Data Science**, 2021. Disponivel em: <https://towardsdatascience.com/understand-the-lyapunov-equation-through-kronecker-product-and-linear-equation-bfff9c1e59ab>. Acesso em: 2023 abr. 2023.

46 TEAM, MATHWORKS STUDENT COMPETITIONS. Student Competition: Mobile Robotics Training. **MathWorks**, 2023. Disponivel em: <https://www.mathworks.com/matlabcentral/fileexchange/62966-student-competition-mobile-robotics-training>. Acesso em: 07 ago. 2023.

47 MATHWORKS TEAM. Simulink and LEGO MINDSTORMS NXT A brief workshop on Simulink Support for Project Based Learning with LEGO MINDSTORMS NXT. **MathWorks**, 2013. Disponivel em: <https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/campaigns/portals/files/project-based-learning/workshop-manual-r2014a.pdf>. Acesso em: 19 abril 2023.

48 MATHWORKS TEAM. Simulink® and LEGO® MINDSTORMS® EV3. **MathWorks**, 2014. Disponivel em: <https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/campaigns/portals/files/project-based-learning/simulink-ev3-workshop-manual-r2015a.pdf>. Acesso em: 3 maio 2023.

49 THE CODING FUN. LEGO MindStorms EV3 – PID Line Follower Code by Using MicroPython 2.0. **The Coding Fun**, 2020. Disponivel em: <https://thecodingfun.com/2020/06/16/lego-mindstorms-ev3-pid-line-follower-code-by-using-micropython-2-0/>. Acesso em: 17 jun. 2023.

50 LOGO FOUNDATION. History of Logo. **Logo Foundation**, 2015. Disponivel em: <https://el.media.mit.edu/logo-foundation/what_is_logo/history.html>. Acesso em: 19 jun. 2023.

51 CONHEÇA o lançamento mundial LEGO® Education SPIKE™ Prime. **Educacional**. Disponivel em: <https://conteudo.site.educacional.com.br/spike-prime-explicando-a-solucao>. Acesso em: 09 Maio 2023.

52 THE LEGO GROUP. O que é o Spike Prime. **LEGO**, 2022. Disponivel em: <https://www.lego.com/pt-br/service/help/spike_prime/about-spike-prime-kA009000001dcNUCAY#:~:text=O%20LEGO®%20Education%20SPIKE,Tudo%20isso%20enquanto%20se%20divertem.>. Acesso em: 09 Maio 2023.

53. K'NEX GROUP. K'nex family learning. **K'NEX GROUP**. Disponivel em: <https://www.knexusergroup.org.uk/en/knex-family-learning.html>. Acesso em: 16 maio 2023.

54. DICIO: DICIONÁRIO ONLINE DE PORTUGUÊS. O que é um robô? **Dicio:** Dicionário Online de Portugûes Disponivel em: <https://www.dicio.com.br/robo/ >. Acesso em: 08 maio 2023.

55. MICHAELLIS: DICIONÁRIO BRASILEIRO DE LÍNGUA PORTUGUESA. Robô. **Michaellis:** Dicionário Brasileiro de Língua Portugues. Disponivel em: <https://michaelis.uol.com.br/moderno-portugues/busca/portugues-brasileiro/robo>. Acesso em: 09 maio 2023.

7 REFERÊNCIAS IMAGENS:

1. PHILO of Alexandria. **Philosophy Basics**. Disponivel em: <https://www.philosophybasics.com/philosophers_philo.html>. Acesso em: 24 ago. 2023.

2. GOVINDARAJAN, M.; MATHWORKS. Obstacle Avoidance using LEGO Mindstorms EV3 and Simulink. **MathWorks**, 2016. Disponivel em: <https://www.mathworks.com/matlabcentral/fileexchange/56293-obstacle-avoidance-using-lego-mindstorms-ev3-and-simulink?s_tid=mwa_osa_a>. Acesso em: 24 jul. 2023

3. OS mitos relacioandos a Leornado da Vinci. **Universo Racionalista**, 2021. Disponivel em: <https://universoracionalista.org/os-mitos-relacionados-a-leonardo-da-vinci/>. Acesso em: 27 abr. 2023.

3. WEG, M. Nikola Tesla. **Museu WEG de Ciência e Tecnologia**, 2015. Disponivel em: <https://museuweg.net/blog/nikola-tesla/>. Acesso em: 27 abr. 2023.

5. VOSS, D.; LEVINE, A. G.; BENNETT-KARASIK, N. This Month in Physics History - February 11, 1738: Jacques de Vaucanson Exhibits Flute-playing Automaton. **Advancing Physics**, 2018. Disponivel em: <https://www.aps.org/publications/apsnews/201802/history.cfm>. Acesso em: 27 abr. 2023.

6. VAZ, M. A. Lego Education Mindstorms Ev3 – 45544. **Nitmaker**. Disponivel em: <https://nitmaker.com.br/produtos/lego-education-mindstorms-ev3-45544/>. Acesso em: 23 ago. 2023.

7. TRANCREDI, S. Leonardo Da Vinci. **Brasil Escola**, Não Informado. Disponivel em: <https://brasilescola.uol.com.br/biografia/leonardo-vinci.htm>. Acesso em: 29 maio 2023.

8. SALLES, A. Tipos de relógios: modelos e características do acessório atemporal. **Segredos do Mundo**, 2023. Disponivel em: <https://segredosdomundo.r7.com/tipos-de-relogios-lista-com-varios-modelos-do-acessorio-atemporal/>. Acesso em: 29 maio 2023.

9. PEARCE, J. Joseph F. Engelberger, a Leader of the Robot Revolution, Dies at 90. **The New York Times**, 2015. Disponivel em: <https://www.nytimes.com/2015/12/03/business/joseph-f-engelberger-a-leader-of-the-robot-revolution-dies-at-90.html>. Acesso em: 27 abr. 2023.

10. O'CONNOR, J. J.; ROBERTSON, E. F. Edward John Routh. **MacTutor**, 2003. Disponivel em: <https://mathshistory.st-andrews.ac.uk/Biographies/Routh/>. Acesso em: 27 abr. 2023.

11. NISE, N. S. **Engenharia de Sistemas de Controle**. 6ª Edição. ed. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora Ltda. Uma editora integrante do GEN | Grupo Editorial Nacional, 2013.

12. NELSON, W.; COX, I. Local Path Control for an Autonomous Vehicle. **Robotics and Automation, IEEE International Conference**, v. 3, p. 1504-1510, 1988.

13. MUIR, P.; NEWMAN, C. Kinematic Modeling for Feedback Control of an Omnidirectional Wheeled Mobile Robot. **Robotics and Automation, IEEE International Conference**, v. 4, p. 1772-1778, 1987.

14. MIRANDA, F. Quem foi Henry Ford e qual a sua importância para a indústria? **SóCientífica**, 2022. Disponivel em: <https://socientifica.com.br/quem-foi-henry-ford-e-qual-a-sua-importancia-para-a-industria/>. Acesso em: 27 abr. 2023.

15. MACIEL, W. Aristóteles. **Infoescola:** Navegando e Aprendendo, Não Informado. Disponivel em: <https://www.infoescola.com/filosofia/aristoteles/>. Acesso em: 29 maio 2023.

16. LOZANO-PÉREZ, T. **Autonomous Robot Vehicles**. [S.l.]: Springer-Verlag, 1990.

17. Nicolas Minorsky. **Wikipedia:** A Enciclopédia Livre, 2021. Disponivel em: <https://pt.wikipedia.org/wiki/Nicolas_Minorsky>. Acesso em: 29 maio 2023.

18. J, S. A PID Controller For Lego Mindstorms Robots. **Inpharmix**, 2018. Disponivel em: <https://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html>. Acesso em: 16 jun. 2023.

19. HANCOCK, J. Evaluating Oscilloscope Sample Rates vs. Sampling Fidelity. **Semantic Scholar**, 2011. Disponivel em: <https://www.semanticscholar.org/paper/Evaluating-Oscilloscope-Sample-Rates-vs.-Sampling-Hancock/0e0c497309ed9a16a6fc298a21dd05c8c3d3180a>. Acesso em: 27 abr. 2023.

20. GROSSMANN, C. Arqueólogos podem ter encontrado túmulo de Aristóteles. **Hypescience**, 2016. Disponivel em: <https://hypescience.com/arqueologos-podem-ter-encontrado-tumulo-de-aristoteles/>. Acesso em: 27 abr. 2023.

21. GOVINDARAJAN, M.; MATHWORKS. Obstacle Avoidance using LEGO Mindstorms EV3 and Simulink. **MathWorks**, 2016. Disponivel em: <https://www.mathworks.com/matlabcentral/fileexchange/56293-obstacle-avoidance-using-lego-mindstorms-ev3-and-simulink?s_tid=mwa_osa_a>. Acesso em: 24 jul. 2023.

22. GIAVITTO, J.-L. **ResearchGate**, 2011. Disponivel em: <https://www.researchgate.net/figure/aucansons-duck-Voltaire-described-Vaucanson-in-these-lines-While-rival-of-the-old_fig1_234144100/download>. Acesso em: 27 abr. 2023.

23. GERDA, A. Quem Criou o Primeiro Robô. **Blog - Loja Roster**, 2023. Disponivel em: <https://www.lojaroster.com.br/blog/quem-criou-o-primeiro-robo-do-mundo/>. Acesso em: 29 maio 2023.

24. FRANKLIN, G. F.; POWELL, J. D.; EMAMI-NAEINI, A. **Feedback Control of Dynamic Systems**. 6ª Edição. ed. [S.l.]: Pearson - Prentice Hall.

25. EVANS, G. W. Bringing root locus to the classroom. **IEEE Xplorre**, 2004. Disponivel em: <https://ieeexplore.ieee.org/document/1368483>. Acesso em: 27 abr. 2023.

26. DOMB, C. James Clerk Maxwell: Scottish mathematician and physicist. **Britannica**, 2023. Disponivel em: <https://www.britannica.com/biography/James-Clerk-Maxwell>. Acesso em: 27 abr. 2023.

27. DIANA, D. Hefesto. **Toda Matéria**. Disponivel em: <https://www.todamateria.com.br/hefesto/>. Acesso em: 25 abr. 2023.

28. DAS, S. et al. Gathering of robots in a ring with mobile faults. **Theoretical Computer Science**, v. 764, p. 42-60, abril 2019. ISSN ISSN: 03043975.

29. DALAKOV, G. Ctesibius of Alexandria. **Computer Timeline**, Não Informado. Disponivel em: <http://www.computer-timeline.com/>. Acesso em: 29 maio 2023.

30. BRYNJOLFSSON, E.; MACFEE, A. **The Second Machine Age:** Work, Progress and Prosperity in a Time of Brillant Technologies. 1ª Edição. ed. New York and London: W.W.Norton & Company, 2014.

31. BOOCH, G. George Devol. **Computer Human Experience**, 2023. Disponivel em: <https://computingthehumanexperience.com/george-devol/>. Acesso em: 27 abr. 2023.

32. BOLTON, W. **Engenharia de Controle**. São Paulo: Makron Books do Brasil Editora Ltda, 1995.

33. BLADIN, P. F. W. Grey Walter, pioneer in the electroencephalogram, robotics, cybernetics, artificial intelligence. **Science Direct**, 2005. Disponivel em: <https://www.sciencedirect.com/science/article/abs/pii/S096758680500398X>. Acesso em: 27 abr. 2023.

34. (陈新宇), X. C. Understanding Lyapunov Equation through Kronecker Product and Linear Equation. **Towards Data Science**, 2021. Disponivel em: <https://towardsdatascience.com/understand-the-lyapunov-equation-through-kronecker-product-and-linear-equation-bfff9c1e59ab>. Acesso em: 2023 abr. 2023.

35. TEAM, MATHWORKS STUDENT COMPETITIONS. Student Competition: Mobile Robotics Training. **MathWorks**, 2023. Disponivel em: <https://www.mathworks.com/matlabcentral/fileexchange/62966-student-competition-mobile-robotics-training>. Acesso em: 07 ago. 2023.

36. MATHWORKS TEAM. Simulink and LEGO MINDSTORMS NXT A brief workshop on Simulink Support for Project Based Learning with LEGO MINDSTORMS NXT. **MathWorks**, 2013. Disponivel em: <https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/campaigns/portals/files/project-based-learning/workshop-manual-r2014a.pdf>. Acesso em: 19 abril 2023.

37. MATHWORKS TEAM. Simulink® and LEGO® MINDSTORMS® EV3. **MathWorks**, 2014. Disponivel em: <https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/campaigns/portals/files/project-based-learning/simulink-ev3-workshop-manual-r2015a.pdf>. Acesso em: 3 maio 2023.

38. SERVO Motor Médio Lego Mindstorms EV3 - 45503 Medium Servo Motor. **wskits - Educativos, Ciência, Robótica e Steam.**, 2023. Disponivel em: <https://www.wskits.com.br/motor-medio-lego-ev3>. Acesso em: 23 ago. 2023.

39. SERVO Motor Grande Lego Mindstorms EV3 - 45502 - Large Servo Motor. **wskits - Educativos, Ciência, Robótica, Steam.**, 23 ago. 2023. Disponivel em: <https://www.wskits.com.br/servo-motor-grande-ev3>.

40. SENSOR Infra Vermelho Robô Lego Mindstorms EV3, 45509 IR Sensor. **wskist-Educativos, Ciência, Robótica e Steam.**, 2023. Disponivel em: <https://www.wskits.com.br/sensor-ir-ev3>. Acesso em: 23 ago. 2023.

41. SENSOR de toque EV3, 45507 EV3 sensor touch para LEGO Mindstorms EV3 Robô. **wskits - Educação, Robótica, Ciência e Steam.**, 2023. Disponivel em: <https://www.wskits.com.br/touch-sensor-ev3>. Acesso em: 23 ago. 2023.

42. SENSOR de Cor e Luz para Robô Lego Mindstorms EV3, Color Sensor EV3. **wskits - Educativos, Ciência, Robótica e Steam.**, 2023. Disponivel em: <https://www.wskits.com.br/sensor-cor-ev3>. Acesso em: 23 ago. 2023.

43. HYPER CULTURA. Quem foi Nikola Tesla? O cientista que sacudiu o mundo com suas inovadoras invenções. **Hiper Cultura**, 2017. Disponivel em: <https://www.hipercultura.com/nikola-tesla/>. Acesso em: 27 abr. 2023.

44. MOINHOS de vento em Amsterdam. **Conexão Amsterdam**, 2020. Disponivel em: <https://www.conexaoamsterdam.com.br/moinhos-de-vento-em-amsterdam/>. Acesso em: 27 jul. 2023.

45. NATIONAL ACADEMIES PRESS. Memorial Tributes: National Academy of Engineering, Volume 3. **National Academy Of Engineering**, 2023. Disponivel em: <https://www.nae.edu/189189/HENDRIK-WADE-BODE-19051982>. Acesso em: 27 abr. 2023.

46. THE CODING FUN. LEGO MindStorms EV3 – PID Line Follower Code by Using MicroPython 2.0. **The Coding Fun**, 2020. Disponivel em: <https://thecodingfun.com/2020/06/16/lego-mindstorms-ev3-pid-line-follower-code-by-using-micropython-2-0/>. Acesso em: 17 jun. 2023.

47. THE EDITORS OF ENCYCLOPAEDIA BRITANNICA. Karel Čapek: Czech writer. **Britannica**, 2023. Disponivel em: <https://www.britannica.com/biography/Karel-Capek>. Acesso em: 27 abr. 2023.

48. POETRIA: POESIA E TEATRO. Homero. **Poetria:** Poesia e Teatro. Disponivel em: <https://poetria.pt/homero>. Acesso em: 27 abr. 2023.

8 APÊNDICES

9 APÊNDICE A – Código de Aquisição de Dados.

```
#!/usr/bin/env pybricks-micropython

# Adição de objetos específicos contidos na biblioteca PyBricks
# Realiza a adição do objeto porta da seção de parâmetros:
# Realiza a adição do objeto motor da seção de dispositivos do EV3:
# Realiza a adição do objeto de datalog, parada de observação e espera da seção de
ferramentas.

from pybricks.ev3devices import Motor,
from pybricks.parameters import Port, Direction
from pybricks.tools import DataLog, StopWatch, wait

# O código apresentado abaixo, cria um arquivo de dato log na pasta onde o projeto
se encontra na memória do Bloco EV3.
# * Por padrão, o ome do arquivo contém o dia e o horário da simulação, por
exemplo:
#   log_2020_02_13_10_07_44_431260.csv
# * Também é possível especificar opcionalmente o título de cada uma das colunas de
dados presentes no arquivo: Por exemplo,
#   Como o objetivo é avaliar o ângulo, a potência, a velocidade linear e
velocidade angular de acordo com o tempo,
#   a construção da variável "data" é realizada da seguinte forma:
data = DataLog('tempo','potencia', 'vel_esq_rad/s', 'vel_dir_rad/s', 'media_rad/s',
'vel_esq_m/s', 'vel_dir_m/s', 'media_m/s',  name='Dados_Motor_Medio_100%',
timestamp=False, extension='csv')
#   Nesse caso, o nome do arquivo foi especificado pela estrutura "name", como
também o seu respectivo tipo pela estrutura "extension"

# Inicialização do motor e início do movimento:
wheel_b = Motor(Port.B,Direction.COUNTERCLOCKWISE,[12,20])
wheel_c = Motor(Port.C,Direction.COUNTERCLOCKWISE,[12,20])
# wheel = Motor(Port.D,Direction.COUNTERCLOCKWISE,[12,36])

# Criação do vetor contendo as diversas potencias utilizadas no motor:
potencia = 0 # inicializa a variável potência

# Estrutura criada utilizando um laço de repetição:
# inicio = -100
# fim = 101
# passo = 5
# potencias = list(range(inicio, fim, passo))
# tam = len(potencias)

potencias = []

# Declaração do vetor e dos valores de potência contidos em cada posição:
# for num in [0, 100, 0, 100, 0, 100, 0, 100, 0, 100, 0, 100, 0, 100]:
```

```
for num in [-100, -95, -90, -85, -80, -75, -70, -65, -60, -55, -50, -45, -40, -35,
30, -25, -15, -10, -5, 0,
           5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90,
95, 100]:
    for i in range(1000):
        potencias.append(num)

# Imprime os valores para verificação:
print(potencias)
# Cácula o tamanho do vetor e armazena o resultado:
tam = len(potencias)
# Imprime o valor para verificação:
print(tam)
# Declaração da variável iterador utilizando a variável "tam" como base para a
criação de uma lista:
iterador = list(range(tam))

# Inicia o temporizador de parada para medir o tempo decorrido:
watch = StopWatch()

# Por segurança o ângulo de início da simulação é estabelecido como zero:
wheel_b.reset_angle(0)
wheel_c.reset_angle(0)

# Simulação do funcionamento do motor com base nas diversas potencias criadas
anteriormente.
for i in iterador:
   potencia = potencias[i]
   wheel_b.dc(potencia)
   wheel_c.dc(potencia)
   P = potencia
   #print(potencia)
   W_b = wheel_b.speed() # velocidade angular em (graus/segundo)
   W_c = wheel_c.speed() # velocidade angular em (graus/segundo)

   W1_b = W_b * (0.0174533) # conversão da velocidade angular em (radianos/segundo)
   W1_c = W_c * (0.0174533) # conversão da velocidade angular em (radianos/segundo)

   m_W1 = (W1_b+W1_c)/2 # (média da velocidade dos dois motores)

   W2_b = W_b/6 # conversão da velocidade angular em (rotações/minuto)
   W2_c = W_c/6 # conversão da velocidade angular em (rotações/minuto)

   S_b = W1_b * 10.2 # cálculo da velocidade linear em (milímetros/segundo)
   S_c = W1_c * 10.2 # cálculo da velocidade linear em (milímetros/segundo)

   S1_b = S_b*0.001 # conversão da velocidade linear em (metros/segundo)
   S1_c = S_c*0.001 # conversão da velocidade linear em (metros/segundo)
   #print(S)
   #print(W)
```

```
   m_S1 = (S1_b+S1_c)/2

   A_b = wheel_b.angle()
   A_c = wheel_c.angle()
   #print(A)

   T = watch.time()
   T = T*0.001
   #print(T)

    # Cada vez que o objeto log() é chamado, uma nova linha é adicionada
    # ao arquivo. Posteiormente, também é possível adiconar mais, caso seja
necessário.
    # Nessa situação, são salvos o tempo atual (time), o ângulo do motor (A), a
potência fornecida ao motor (P),
    # A velocidade linear (S), e velocidade angular (W):
   data.log(T, P, W1_b, W1_c, m_W1, S1_b, S1_c, m_S1)

    # Espera um segundo para que o motor possa se movimentar um pouco com a
potência fornecida:
   wait(10)
```

10 **APÊNDICE B –** Código de tratamento de dados coletados pelo sensor.

```
% Importação dos Dados Presentes no Arquivo (.xlsx)
% Carregamento dos dados referentes ao tempo de simulação:
time = xlsread("Dados_Motor_Direito_100%.xlsx", "A2:A14001");

% Carregameento dos dados referentes aos ângulos assumidos pelo motor:
angulo = xlsread("Dados_Motor_Direito_100%.xlsx", "B2:B14001");

% Carregamento dos dados referentes a potência fornecida ao motor:
potencia = xlsread("Dados_Motor_Direito_100%.xlsx", "C2:C14001");

% Carregamentos dos dados referentes a velocidade assumida pelo motor:
velocidadeL = xlsread("Dados_Motor_Direito_100%.xlsx", "D2:D14001");
velocidadeA = xlsread("Dados_Motor_Direito_100%.xlsx", "F2:F14001");
velocidadeR = xlsread("Dados_Motor_Direito_100%.xlsx", "H2:H14001");

% Inicialização:
pwm2voltage = 1; % Fator de Conversão entre PWM e Tensão elétrica

% Remove os dois pontos iniciais dos dados (Geralmente ruidoso):
time = time(3:end);
angulo = angulo(3:end);
potencia = potencia(3:end);
velocidadeL = velocidadeL(3:end);
velocidadeA = velocidadeA(3:end);
velocidadeR = velocidadeR(3:end);

% Cria um filtro passa-baixas IIR de segunda ordem:
d1 = designfilt('lowpassiir','DesignMethod','butter', ...
                'FilterOrder',2,'HalfPowerFrequency',3/25);

% Filtra os sinais adquiridos no DataLogging originados pelo Motor:
angulo = filter(d1,angulo);
velocidadeL = filter(d1,velocidadeL);
velocidadeA = filter(d1,velocidadeA);
velocidadeR = filter(d1,velocidadeR);

% Divide uma parcela dos dados adquiridos para estimar os parâmetros:
indexHalf = ceil(length(time)/2);
% Divide uma parcela dos dados adquiridos para validar os parâmetros:
timeEst = time(1:indexHalf);

% Dados dedicados para estimar os parâmetros:
inputsEst = pwm2voltage.* potencia(1:indexHalf);
outputsEst = velocidadeA(1:indexHalf);
outputsEst1 = velocidadeR(1:indexHalf);

% Dados dedicados para validar os parâmetros:
inputsVal = pwm2voltage.* potencia(indexHalf+1:end);
timeVal = time(1:length(inputsVal));
outputsVal = velocidadeA(indexHalf+1:end);
outputsVal1 = velocidadeR(indexHalf+1:end);

% Define os valores iniciais de cada parâmetro:
b = 1;
J = 1;
k = 1;
```

```
La = 1;
Ra = 1;
```

11 **APÊNDICE C** – Código de leitura do valor da intensidade luminosa da superfície em porcentagem.

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile



# Cria o objeto relacionado ao bloco Ev3
ev3 = EV3Brick()

# Cria o objeto relacionado ao sensor de cor e associa a porta utilizada no bloco
Ev3
luz = ColorSensor(Port.S1)

# Simulação do Programa:
# Cada 10 milissegundos uma leitura é feita para computar a porcentagem da
intensidade
# da superfície onde o sensor foi posicionado.
while True:
    # Criação da variável porcentagem para armazenar o valores de cada leitura.
    porcentagem = luz.reflection()
    # Impressão da leitura atual na tela do terminal do computador.
    print(porcentagem)
    # Espera 10 milissegundos para realizar uma nova leiitura.
    wait(10)
```

12 **APÊNDICE D –** Controlador ON/OFF.

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# O código apresentado abaixo, cria um arquivo de dato log na pasta onde o projeto
se encontra na memória do Bloco EV3.
# * Por padrão, o nome do arquivo contém o dia e o horário da simulação, por
exemplo:
#   log_2020_02_13_10_07_44_431260.csv
# * Também é possível especificar opcionalmente o título de cada uma das colunas de
dados presentes no arquivo: Por exemplo,
#   Como o objetivo é avaliar o ângulo, a potência, a velocidade linear e
velocidade angular de acordo com o tempo,
#   a contrução da variável "data" é realizada da seguinte forma:
data = DataLog('tempo','erro', name='Seguidor de Linha ON OFF', timestamp=False,
extension='csv')
#   Nesse caso, o nome do arquivo foi especificado pela estrutura "name", como
também o seu respectivo tipo pela estrutura "extension"

# Inicializa dois objetos do tipo motor com no sentido anti-horário:
mesquerdo = Motor(Port.B, Direction.COUNTERCLOCKWISE)
mdireito = Motor(Port.C, Direction.COUNTERCLOCKWISE)

# Inicializa objeto sensor de luz na porta S1.
sluz = ColorSensor(Port.S1)

# Variáveis de Armazenamento
vleitura = []
verro = []

# # Calcula o limiar baseado nos valores atribuídos as cores branca e preta.
preto = 5
branco = 50
limiar = (preto + branco) / 2

# Estabelece a velocidade em 200 milímetros por segundo.
DRIVE_SPEED = 200

# Inicializa a avaliação do comportamento do robô.
watch = StopWatch()

while True:
   # Realiza a leitura da intensidade luminosa conduzida pelo sensor.
```

```
   leitura = sluz.reflection()
   # Constroí uma lista com os valores de cada leitura conduzida pelo sensor.
   vleitura.append(leitura)

   # Calcula o valor do erro subtraindo a leitura atual do limiar.
   erro = sluz.reflection() - limiar
   # Constroí uma lista com os valores de cada erro calculado.
   verro.append(erro)

   # Caso a leitura seja menor ou igual ao limiar, o robô vira à esquerda
   if leitura <= limiar:
      mesquerdo.run_angle(DRIVE_SPEED,50)
      mdireito.stop()

   # Caso contrário, o robô vira à direita
   else:
      mdireito.run_angle(DRIVE_SPEED,50)
      mesquerdo.stop()

   # Estabelece a avaliação do tempo decorrido durante a simulação.
   T = watch.time()
   T = T*0.001

   # Computa os valores do tempo, do erro e da curva em uma linha da tabela de
dados.
   data.log(T, erro)
   # Espera 10 milissegundos para a realização da próxima leitura.
   wait(10)
```

13 **APÊNDICE E –** Controlador 3 Níveis.

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# O código apresentado abaixo, cria um arquivo de dato log na pasta onde o projeto
se encontra na memória do Bloco EV3.
# * Por padrão, o nome do arquivo contém o dia e o horário da simulação, por
exemplo:
#   log_2020_02_13_10_07_44_431260.csv
# * Também é possível especificar opcionalmente o título de cada uma das colunas de
dados presentes no arquivo: Por exemplo,
#   Como o objetivo é avaliar o ângulo, a potência, a velocidade linear e
velocidade angular de acordo com o tempo,
#   a contrução da variável "data" é realizada da seguinte forma:
data = DataLog('tempo','error', name='Seguidor de Linha 3 Niveis', timestamp=False,
extension='csv')
#   Nesse caso, o nome do arquivo foi especificado pela estrutura "name", como
também o seu respectivo tipo pela estrutura "extension"

# Inicializa dois objetos do tipo motor com no sentido anti-horário:
mesquerdo = Motor(Port.B, Direction.COUNTERCLOCKWISE)
mdireito= Motor(Port.C, Direction.COUNTERCLOCKWISE)

# Inicializa objeto sensor de luz na porta S1.
sluz = ColorSensor(Port.S1)

# Variáveis de Armazenamento
vleitura = []
verro = []

# Calcula o limiar baseado nos valores atribuídos as cores branca e preta.
preto = 5
branco = 50
limiar1 = int((branco - preto/3)+preto
limiar2 = branco - int((branco - preto)/3)

# Estabelece a velocidade em 200 milimetros por segundo.
DRIVE_SPEED = 200

# Inicializa a avaliação do comportamento do robô.
watch = StopWatch()

while True:
```

```
  # Realiza a leitura da intensidade luminosa conduzida pelo sensor.
  leitura = sluz.reflection()
  # Constroí uma lista com os valores de cada leitura conduzida pela sensor.
  vleitura.append(leitura)

  # Caso a leitura seja menor ou igual ao limiar1, o robô fará uma curva à direita.
  if leitura <= limiar1:
    mdireito.run_angle(DRIVE_SPEED,50)
    mesquerdo.stop()
    erro = sluz.reflection() - limiar1
    verro.append(erro)

  # Caso a leitura seja maior ou igual ao limiar2, o robô fará uma curva à
esquerda.
  elif leitura >= limiar2:
    mesquerdo.run_angle(DRIVE_SPEED,50)
    mdireito.stop()
    erro = sluz.reflection() - limiar2
    verro.append(erro)

  # Caso contrário o robô seguirá em frente
  else:
    mdireito.run(DRIVE_SPEED)
    mesquerdo.run(DRIVE_SPEED)
    erro = 0
    verro.append(erro)

  # Estabelece a avaliação do tempo decorrido durante a simulação.
  T = watch.time()
  T = T*0.001

  # Computa os valores do tempo, do erro e da curva em uma linha da tabela de
dados.
  data.log(T, erro)

  # Espera 10 milissegundos para a realização da próxima leitura.
  wait(10)
```

14 **APÊNDICE F –** Controlador P.

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# O código apresentado abaixo, cria um arquivo de dato log na pasta onde o projeto
se encontra na memória do Bloco EV3.
# * Por padrão, o nome do arquivo contém o dia e o horário da simulação, por
exemplo:
#   log_2020_02_13_10_07_44_431260.csv
# * Também é possível especificar opcionalmente o título de cada uma das colunas de
dados presentes no arquivo: Por exemplo,
#   Como o objetivo é avaliar o ângulo, a potência, a velocidade linear e
velocidade angular de acordo com o tempo,
#   a contrução da variável "data" é realizada da seguinte forma:
data = DataLog('tempo','erro','curva',name='Seguidor de Linha P', timestamp=False,
extension='csv')
#   Nesse caso, o nome do arquivo foi especificado pela estrutura "name", como
também o seu respectivo tipo pela estrutura "extension"

# Inicializa dois objetos do tipo motor com no sentido anti-horário:
mesquerdo = Motor(Port.B, Direction.COUNTERCLOCKWISE)
mdireito= Motor(Port.C, Direction.COUNTERCLOCKWISE)

# Inicializa objeto sensor de luz na porta S1.
sluz = ColorSensor(Port.S1)

# Variáveis de Armazenamento
vleitura = []
verro = []

# Calcula o limiar baseado nos valores atribuídos as cores branca e preta.
preto = 5
branco = 50
limiar = (preto + branco)/2

# Estabelece a velocidade em 200 milimetros por segundo.
DRIVE_SPEED = 200

# Estabelece o valor do ganho proporcional.
PROPORTIONAL_GAIN = -5

# Inicializa a avaliação do comportamento do robô.
watch = StopWatch()
```

```
while True:
    # Realiza a leitura da intensidade luminosa conduzida pelo sensor.
    leitura = sluz.reflection()
    # Constroí uma lista com os valores de cada leitura conduzida pela sensor.
    vleitura.append(leitura)

    # Calcula o valor do erro subtraindo a leitura atual do limiar.
    erro = sluz.reflection() - limiar
    # Constroí uma lista com os valores de cada erro calculado.
    verro.append(erro)

    # Estabelece a avaliação do tempo decorrido durante a simulação.
    T = watch.time()
    T = T*0.001

    # Calcula o valor a ser atrubuído durante a realização de uma curva.
    curva = (PROPORTIONAL_GAIN * erro)

    # Computa os valores do tempo, do erro e da curva em uma linha da tabela de
dados.
    data.log(T, erro, curva)

    # Atribuí o valor da curva em conjunto com a velocidade desenvolvida pelos
motores direito e esquerdo.
    mdireito.run(int(DRIVE_SPEED-curva))
    mesquerdo.run(int(DRIVE_SPEED+curva))

    # Espera 10 milissegundos para a realização da próxima leitura.
    wait(10)
```

15 **APÊNDICE G –** Controlador PI.

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# O código apresentado abaixo, cria um arquivo de dato log na pasta onde o projeto
se encontra na memória do Bloco EV3.
# * Por padrão, o nome do arquivo contém o dia e o horário da simulação, por
exemplo:
#   log_2020_02_13_10_07_44_431260.csv
# * Também é possível especificar opcionalmente o título de cada uma das colunas de
dados presentes no arquivo: Por exemplo,
#   Como o objetivo é avaliar o ângulo, a potência, a velocidade linear e
velocidade angular de acordo com o tempo,
#   a contrução da variável "data" é realizada da seguinte forma:
data = DataLog('tempo','erro','integral', 'curva',name='Seguidor de Linha PI',
timestamp=False, extension='csv')
#   Nesse caso, o nome do arquivo foi especificado pela estrutura "name", como
também o seu respectivo tipo pela estrutura "extension"

# Inicializa dois objetos do tipo motor com no sentido anti-horário:
mesquerdo = Motor(Port.B, Direction.COUNTERCLOCKWISE)
mdireito= Motor(Port.C, Direction.COUNTERCLOCKWISE)

# Inicializa objeto sensor de luz na porta S1.
sluz = ColorSensor(Port.S1)

# Variáveis de Armazenamento
vleitura = []
verro = []

# Calcula o limiar baseado nos valores atribuídos as cores branca e preta.
preto = 5
branco = 50
limiar = (preto + branco)/2

# Estabelece a velocidade em 200 milimetros por segundo.
DRIVE_SPEED = 200

# Estabelece o valor do ganho proporcional e do ganho integral.
PROPORTIONAL_GAIN = -5
INTEGRAL_GAIN = 1

integral = 0
```

```
# Inicializa a avaliação do comportamento do robô.
watch = StopWatch()

while True:
    # Realiza a leitura da intensidade luminosa conduzida pelo sensor.
    leitura = sluz.reflection()
    # Constroí uma lista com os valores de cada leitura conduzida pela sensor.
    vleitura.append(leitura)

    # Calcula o valor do erro subtraindo a leitura atual do limiar.
    erro = sluz.reflection() - limiar
    # Constroí uma lista com os valores de cada erro calculado.
    verro.append(erro)

    if integral <= 0:
    # Atribuí o erro associado a leitura atual na parcel integrativa do
controlador.
        integral = (1/2)*integral + erro

    else:
        integral = 0

    # Estabelece a avaliação do tempo decorrido durante a simulação.
    T = watch.time()
    T = T*0.001

    # Calcula o valor a ser atrubuído durante a realização de uma curva.
    curva = (PROPORTIONAL_GAIN * erro) + (INTEGRAL_GAIN * integral)

    # Computa os valores do tempo, do erro, da integral e da curva em uma linha da
tabela de dados.
    data.log(T, erro, integral, curva)

    # Atribuí o valor da curva em conjunto com a velocidade desenvolvida pelos
motores direito e esquerdo.
    mdireito.run(int(DRIVE_SPEED-curva))
    mesquerdo.run(int(DRIVE_SPEED+curva))

    # Espera 10 milissegundos para a realização da próxima leitura.
    wait(10)
```

16 **APÊNDICE H –** Controlador PD.

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# O código apresentado abaixo, cria um arquivo de dato log na pasta onde o projeto
se encontra na memória do Bloco EV3.
# * Por padrão, o nome do arquivo contém o dia e o horário da simulação, por
exemplo:
#   log_2020_02_13_10_07_44_431260.csv
# * Também é possível especificar opcionalmente o título de cada uma das colunas de
dados presentes no arquivo: Por exemplo,
#   Como o objetivo é avaliar o ângulo, a potência, a velocidade linear e
velocidade angular de acordo com o tempo,
#   a contrução da variável "data" é realizada da seguinte forma:
data = DataLog('tempo','erro','curva',name='Seguidor de Linha PD', timestamp=False,
extension='csv')
#   Nesse caso, o nome do arquivo foi especificado pela estrutura "name", como
também o seu respectivo tipo pela estrutura "extension"

# Inicializa dois objetos do tipo motor com no sentido anti-horário:
mesquerdo = Motor(Port.B, Direction.COUNTERCLOCKWISE)
mdireito= Motor(Port.C, Direction.COUNTERCLOCKWISE)

# Inicializa objeto sensor de luz na porta S1.
sluz = ColorSensor(Port.S1)

# Variáveis de Armazenamento
vleitura = []
verro = []

# Calcula o limiar baseado nos valores atribuídos as cores branca e preta.
preto = 5
branco = 50
limiar = (preto + branco)/2

# Estabelece a velocidade em 200 milimetros por segundo.
DRIVE_SPEED = 200

# Estabelece o valor do ganho proporcional.
PROPORTIONAL_GAIN = -5
DERIVATIVE_GAIN = 0.0001

# Inicializa a avaliação do comportamento do robô.
```

```
watch = StopWatch()

# Avaliação do erro
derivativo = 0
ultimo_erro = 0

# Start following the line endlessly.
while True:
    # Realiza a leitura da intensidade luminosa conduzida pelo sensor.
    leitura = sluz.reflection()
    # Constroí uma lista com os valores de cada leitura conduzida pela sensor.
    vleitura.append(leitura)

    # Calcula o valor do erro subtraindo a leitura atual do limiar.
    erro = sluz.reflection() - limiar
    # Constroí uma lista com os valores de cada erro calculado.
    verro.append(erro)

    derivativo = erro - ultimo_erro

    # Estabelece a avaliação do tempo decorrido durante a simulação.
    T = watch.time()
    T = T*0.001

    # Calcula o valor a ser atrubuído durante a realização de uma curva.
    curva = (PROPORTIONAL_GAIN * erro) + (DERIVATIVE_GAIN * derivativo)

    # Computa os valores do tempo, do erro e da curva em uma linha da tabela de
dados.
    data.log(T, erro, derivativo, curva)

    # Atribuí o valor da curva em conjunto com a velocidade desenvolvida pelos
motores direito e esquerdo.
    mdireito.run(int(DRIVE_SPEED-curva))
    mesquerdo.run(int(DRIVE_SPEED+curva))
    ultimo_erro = erro

    # Espera 10 milissegundos para a realização da próxima leitura.
    wait(10)
```

17 **APÊNDICE I –** Controlador PID.

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# O código apresentado abaixo, cria um arquivo de dato log na pasta onde o projeto
se encontra na memória do Bloco EV3.
# * Por padrão, o nome do arquivo contém o dia e o horário da simulação, por
exemplo:
#   log_2020_02_13_10_07_44_431260.csv
# * Também é possível especificar opcionalmente o título de cada uma das colunas de
dados presentes no arquivo: Por exemplo,
#   Como o objetivo é avaliar o ângulo, a potência, a velocidade linear e
velocidade angular de acordo com o tempo,
#   a contrução da variável "data" é realizada da seguinte forma:
data = DataLog('tempo','erro', 'integral', 'derivada', 'curva',name='Controlador
PID', timestamp=False, extension='csv')
#   Nesse caso, o nome do arquivo foi especificado pela estrutura "name", como
também o seu respectivo tipo pela estrutura "extension"

# Inicializa dois objetos do tipo motor com no sentido anti-horário:
mesquerdo = Motor(Port.B, Direction.COUNTERCLOCKWISE)
mdireito= Motor(Port.C, Direction.COUNTERCLOCKWISE)

# Inicialização do sensor de luz na porta S1:
sluz = ColorSensor(Port.S1)

# Variáveis de Armazenamento:
vleitura = []
verro = []

# Calcula o limiar baseado nos valores atribuídos as cores branca e preta:
preto = 5
branco = 50
limiar = (preto + branco) / 2

# Estabelece a velocidade em 200 milimetros por segundo.
DRIVE_SPEED = 200

# Estabelece o valor do ganho proporcional, integral e integrativo.
PROPORTIONAL_GAIN = -6
INTEGRAL_GAIN = 1
DERIVATIVE_GAIN = 0.1
```

```
# Inicialização das variáveis de auxiliares no cálculo dos ganhos:
integral = 0
derivativo= 0
ultimo_erro = 0


watch = StopWatch()

# Start following the line endlessly.
while True:
    # Realiza a leitura da intensidade luminosa conduzida pelo sensor.
    leitura = sluz.reflection()
    # Constrói uma lista com os valores de cada leitura conduzida pela sensor.
    vleitura.append(leitura)

    # Calcula o valor do erro subtraindo a leitura atual do limiar.
    erro = sluz.reflection() - limiar
    # Constrói uma lista com os valores de cada erro calculado.
    verro.append(erro)

    # Composição de erro avaliando o termo integral e o termo derivativo.
    if integral <= 0:
    # Atribuí o erro associado a leitura atual na parcela integrativa do
controlador.
        integral = (1/2)*integral + erro

        if integral > 0:
            integral = 0

    else:
        integral = 0

    derivativo = erro - ultimo_erro

    # Estabelece a avaliação do tempo decorrido durante a simulação.
    T = watch.time()
    T = T*0.001

    # Cálculo de curva em função de cada um dos termos:
    curva = (PROPORTIONAL_GAIN * erro) + (INTEGRAL_GAIN * integral) +
(DERIVATIVE_GAIN * derivativo)
    # Contabilização dos valores na tabela:
    data.log(T, erro, integral, derivativo, curva)

    # Atribuí o valor da curva em conjunto com a velocidade desenvolvida pelos
motores direito e esquerdo.
    mdireito.run(int(DRIVE_SPEED+curva))
    mesquerdo.run(int(DRIVE_SPEED-curva))
    ultimo_erro = erro
```

```
    # Espera 10 milissegundos para a realização da próxima leitura.
    wait(10)
```

18 **APÊNDICE J –** Construção da função de transferência dos motores.

```
% Construção da Função de Transferência Utilizando os Valores Obtidos Por
% Meio da Simulação dos Valores de DataLog Computados Em Relação Ao Motor
% Direito e Em Relação Ao Motor Esquerdo:

% Atribuição Dos Valores Associados A Resistência Elétrica de Armadura
% (Ra), A Indutância Elétrica de Armadura (La), A Constante de Torque ou A
% Constante de Tensão Elétrica de Retorno (K), O Momento de Inércia (J) e O
% Coeficiente de Amortecimento:

% Motor Direito:
Ra_Motor_Direito = 0.45827;
La_Motor_Direito = 0.066444;
K_Motor_Direito = 0.41247;
J_Motor_Direito = 0.066444;
b_Motor_Direito = 0.45827;

% Constantes Função de Transferência do Motor Direito:
num_direito = 8*(K_Motor_Direito/(J_Motor_Direito*La_Motor_Direito));
d0_direito = ((b_Motor_Direito*Ra_Motor_Direito)+(K_Motor_Direito* ...
    K_Motor_Direito))/(J_Motor_Direito*La_Motor_Direito);
d1_direito = ((b_Motor_Direito*La_Motor_Direito)+ ...
    (J_Motor_Direito*Ra_Motor_Direito))/(J_Motor_Direito*La_Motor_Direito);
d2_direito = 1;

% Motor Esquerdo:
Ra_Motor_Esquerdo = 0.50886;
La_Motor_Esquerdo = 0.069281;
K_Motor_Esquerdo = 0.45089;
J_Motor_Esquerdo = 0.069281;
b_Motor_Esquerdo = 0.50886;

% Constantes Função de Transferência do Motor Esquerdo:
num_esquerdo = 8*(K_Motor_Esquerdo/(J_Motor_Esquerdo*La_Motor_Esquerdo));
d0_esquerdo = ((b_Motor_Esquerdo*Ra_Motor_Esquerdo)+(K_Motor_Esquerdo* ...
    K_Motor_Esquerdo))/(J_Motor_Esquerdo*La_Motor_Esquerdo);
d1_esquerdo = ((b_Motor_Esquerdo*La_Motor_Esquerdo)+ ...

(J_Motor_Esquerdo*Ra_Motor_Esquerdo))/(J_Motor_Esquerdo*La_Motor_Esquerdo);
d2_esquerdo = 1;

% Função de Transferência do Motor Direito no Domínio do Tempo Contínuo:
motor_direito = tf(num_direito,[d2_direito, d1_direito, d0_direito]);
% Conversão para o domínio de tempo discreto:
Ts = 0.01;
disc_md = c2d(motor_direito, Ts);

% Função de Transferência do Motor Esquerdo no Domínio do Tempo Contínuo:
motor_esquerdo = tf(num_esquerdo,[d2_esquerdo, d1_esquerdo, d0_esquerdo]);
% Conversão para o domínio de tempo discreto:
Ts = 0.01;
disc_me = c2d(motor_esquerdo, Ts);

A = [-Ra_Motor_Direito/La_Motor_Direito -K_Motor_Direito/La_Motor_Direito;
    K_Motor_Direito/J_Motor_Direito -b_Motor_Direito/J_Motor_Direito];
B = [K_Motor_Direito/(J_Motor_Direito*La_Motor_Direito); 0];
C = [0 1];
```

```
D = [0];

sys_dc = ss(A,B,C,D);
sys_tf = tf(sys_dc);
```

19 APÊNDICE K – Construção da Tabela 1-D.

```
% Carrega as informações adquiridas dos motores direito e esquerdo criando
% uma função de primeiro grau responsável pela construção da relação velo-
% cidade e potência do motor no diagrama de blocos.
time = xlsread("Tabela_1D_1.xlsx", "A2:A20501");
pot_motor_direito = xlsread("Tabela_1D_1.xlsx","J2:J205001");
vel_motor_direito = xlsread("Tabela_1D_1.xlsx","L2:L205001");
pot_motor_esquerdo = xlsread("Tabela_1D_1.xlsx","J2:J205001");
vel_motor_esquerdo = xlsread("Tabela_1D_1.xlsx","K2:K205001");
```