UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Filipe Peticor Mei

Movimentação de um Robô Uniciclo Utilizando Controladores PID Discretos

Recife

2020

Filipe Peticor Mei

Movimentação de um Robô Uniciclo Utilizando Controladores PID Discretos

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Estatística e Informática

Curso de Bacharelado em Sistemas de Informação

Orientador: Victor Wanderley Costa de Medeiros

Coorientador: Glauco Estácio Gonçalves

Recife

2020

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

M499m Mei, Filipe Peticor
Movimentação de um Robô Uniciclo Utilizando Controladores PID Discretos / Filipe Peticor Mei. - 2020.
48 f. : il.

Orientador: Victor Wanderley Costa De Medeiros.
Coorientador: Glauco Estacio Goncalves.
Inclui referências.

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Sistemas da Informação, Recife, 2021.

1. Controle PID. 2. Arduino. 3. Robótica. 4. Paradigma Reativo. 5. Controle de Trajetória. I. Medeiros, Victor Wanderley Costa De, orient. II. Goncalves, Glauco Estacio, coorient. III. Título

CDD 004

FILIPE PETICOR MEI

MOVIMENTAÇÃO DE UM ROBÔ UNICICLO UTILIZANDO CONTROLADORES PID DISCRETOS

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada em: 20 de Julho de 2021.

BANCA EXAMINADORA

Victor Wanderley Costa De Medeiros
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco

Glauco Estácio Gonçalves
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco

Cícero Garrozi
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco

À minha família, amigos, professores e todos que, direta ou indiretamente, contribuíram para minha formação…

Agradecimentos

Agradeço à minha família, em especial meu pai e minha mãe, Gilberto e Rosana Mei, pela educação que me deram, ensinamentos e integridade. Se não fosse eles, não teria a oportunidade de estudar em um curso de ensino superior e enfrentar as adversidades da vida, moldando a pessoa que sou hoje.

Agradeço à minha noiva e futura esposa Clara Wanderley, que está comigo desde terceiro período do curso até a sua finalização deste trabalho. Me acompanhando durante os momentos mais difíceis, me ajudando a me manter calmo e focado, mesmo não sendo sua responsabilidade. Sua ajuda foi muito importante para este trabalho, também possuindo mérito na conclusão desse documento.

Agradeço aos colegas e amigos que fiz na universidade, em especial Airton, Daniel e Manoel, que me ajudaram desde problemas pessoais a projetos de cadeiras. Construindo uma amizade que vai além do curso.

Agradeço à universidade e a todos os professores da UFRPE, pela oportunidade que foi estudar no curso de Sistemas de Informação, moldando a visão profissional que tenho hoje.

Por fim, agradeço os meus orientadores, Prof. Glauco Gonçalves e Victor Medeiros, por toda paciência e orientação durante o curso. Me mostrando desde a iniciação científica ate a escrita deste documento o que a universidade tem a oferecer na vida pessoal e profissional de uma pessoa.

“O insucesso é apenas uma oportunidade para recomeçar com mais inteligência.”
(Henry Ford)

Resumo

Robôs autônomos são robôs que podem realizar atividades e atingir objetivos em ambientes desestruturados. Um dos principais objetivos desse tipo de equipamento é realizar atividades que são consideradas repetitivas, perigosas ou inviáveis para o ser humano, sendo utilizados em ambiente doméstico ou industriais. Para atingir esse objetivo é essencial que o robô autônomo tenha um bom sistema de localização. Variados tipos de algoritmos de sistemas controle de trajetória são propostos e o controlador PID é um dos controladores clássicos populares em diversas aplicações, sendo possível usar esse controlador para a finalidade de sistemas de navegação. Dessa forma, este trabalho propõe o desenvolvimento de um protótipo de um robô, para avaliar como um controlador PID digital auxilia o desempenho de sua movimentação.

Palavras-chave: Controle PID, arduino, robótica, paradigma reativo, controle de trajetória.

Abstract

Autonomus robots are those who are able to complet tasks and goals in unstructured enviroments. One of the main goals of these type of equipment is to complet tasks that are considered to be repetitive, dangerous or impracticable to humans, beign used in domestic or industrial environments. To reach that goal, it's essencial that the autonomous robot has a good location system. Various types of trajectory system control algorithms are proposed and the PID control is one of the classical controls that are popular in many aplications, making it possible to use it as a navigating system. Therefore, this reasearch proposes the development of a robot prototype, to asses how a digital control PID assists the development of it's movement.

Keywords: PID controller, arduino, robotics, reactive paradigm, trajectory control

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

PID	*Proportional, Integral and Derivative* (Proporcional, Integral e Derivativo)
FPGA	*Field-Programmable Gate Array* (Arranjo de Portas Programáveis em Campo)
RFID	*Radio-Frequency Identification* (Identificação por Radiofrequência)
SP	*Setpoint* (Valor-alvo)
MPU	*Motion Processing Unit* (Unidade de Processamento de Movimentos)
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
SPA	*Sense, Plan and Act* (Sentir, Planejar e Agir)
SA	*Sense and Act* (Sentir e Agir)
PSA	*Plan, Sense and Act* (Planejar, Sentir e Agir)
USB	*Universal Serial Bus* (Porta Serial Universal)
DC	*Direct Current* (Corrente Contínua)
DMP	*Digital Motion Processor* (Processador de Movimento Digital)
RAM	*Random Access Memories* (Memória de Acesso Aleatório)
EEPROM	*Electrically Erasable Programmable Read-Only Memory* (Memória Somente de Leitura Programável Apagável Eletricamente).
RISC	*Reduced Instruction Set Computer* (Computador com um Conjunto Reduzido de Instruções)
DoF	*Degrees of Freedom* (Graus de Liberdade)

Sumário

1 Introdução e Motivação

Robôs autônomos são robôs que podem realizar atividades e atingir objetivos em ambientes desestruturados, podendo trabalhar por um prolongado período de tempo, se deslocar de um ponto A a um ponto B e receber informações do seu ambiente sem a intervenção humana (MEDEIROS, 1998).

Um dos principais objetivos desse tipo de equipamento é realizar atividades que são consideradas repetitivas, perigosas ou inviáveis para o ser humano. Podem ser utilizados na agricultura como colheitadeiras automáticas (SANTOS et al., 2020), em hospitais com robôs cirúrgicos (SELIM; EL-AMARY; DAHAB, 2012), em fábricas na linha de produção com soldagem feita por robôs (WANG et al., 2012), entre outros usos. No uso doméstico podem ser utilizados em ambientes internos, como aspiradores de pó inteligentes (YASUTOMI; YAMADA; TSUKAMOTO, 1988), e em ambientes externos, como cortadores de grama (BOSSE; NOURANI-VATANI; ROBERTS, 2007).

Para atingir esse objetivo é essencial que o robô autônomo tenha um bom sistema de localização, consiga perceber o ambiente em tempo real e consiga manter sua integridade, utilizando seus sensores e sistemas embarcados. Obviamente, usando essas informações, é muito difícil construir um modelo preciso e completo em tempo real para o planejamento do percurso.

Variados tipos de algoritmos de sistemas controle de trajetória são propostos, como a técnica do integrador de um passo atrás (*backstepping*) (FIERROR, 1997), sistemas de controle utilizando lógica difusa (*fuzzy logic*) (EL-TELEITY et al., 2011) e redes neurais (KUMAR et al., 2015). Em contrapartida, controladores Proporcional Integral Derivativo (PID) são mais simples de implementar e é uma conhecida estratégia de controle em ambientes industriais, por sua simplicidade de utilização e atuação satisfatória, se comparado com outros tipos de controle, esta pode ser considerada umas das melhores estratégias para controle em ambientes industriais (BING; HUA, 2011).

Na verdade, o controlador PID é um dos controladores clássicos populares em diversas aplicações, sendo possível usar esse controlador para a finalidade de sistemas de navegação de robôs autônomos, uma vez que o uso do PID promete muitas vantagens como precisão, suavidade e desempenho.

Porém, um dos desafios de um controlador PID é obter os valores ótimos de seus parâmetros. Mesmo que estratégias automáticas tenham sido propostas para ajustar esses parâmetros, raramente essas estratégias são aplicadas devido à sua complexidade (ANG; CHONG; LI, 2005).

1.1 Trabalhos Relacionados

O presente capítulo apresenta uma síntese dos trabalhos relacionados a este que fazem uso de técnicas de controle PID discreto na movimentação de robôs. O objetivo é mostrar uma visão geral do uso da teoria de controle, diferentes abordagens e a melhoria de eficiência, precisão e confiabilidade que o uso desse tipo técnica de controle traz.

Em Marosan e Constantin (2020) é desenvolvido um sistema de movimentação de um robô utilizando um controlador PID discreto e um sensor de giroscópio. Nos casos de testes os autores buscam a viabilidade do uso desse tipo de sensor como sinal de referência para o controlador. Foram realizados três casos de teste, em que todos o robô realizava uma curva a um ângulo em seu próprio eixo. No primeiro teste foi utilizado um controle direto, para avaliar o controle de movimentação sem o controlador PID, e nos testes seguintes um controlador PID discreto com diferentes configurações dos seus parâmetros de ganho. Não foi utilizado nenhum método de sintonia dos parâmetros do controlador PID, sendo realizada de forma empírica. A conclusão do trabalho é que o giroscópio pode ser utilizado como sinal referência para esse tipo de sistema, melhorando significativamente a execução e acurácia do movimento.

No trabalho de (MENG et al., 2018) também é desenvolvido um sistema de movimentação utilizando o controlador PID discreto. Diferente do trabalho anterior, neste foi utilizado um controlador para cada roda do robô, utilizando *encoders* (codificadores) rotativos como sinal de referência, totalizando dois controladores no sistema. Foram utilizados dois casos de testes, onde no primeiro o robô andava em linha reta e no segundo realizava um movimento no formato de S. Durante o experimento o robô realizou as movimentações como esperado e o uso dos controladores PID nas rodas se mostrou eficiente, conseguindo manter a velocidade das rodas durante as movimentações no valor desejado. O método de sintonia dos parâmetros do controlador PID utilizado foi o Ziegler-Nichols (ZIEGLER; NICHOLS et al., 1942).

No estudo de (BÂRSAN, 2019) foi utilizado um robô com duas rodas, cada uma com seu próprio motor. Diferente dos outros estudos, esse robô possui rodas com diâmetro diferentes, tornando a movimentação do robô imprecisa. O caso de teste envolvia o robô realizar um movimento de ida e volta em um espaço, como referência para a movimentação foi utilizado os *encoders* (codificadores) rotativos que cada roda possuía. O controlador PID foi implementado e ajustado utilizando os softwares MATLAB[1] e Simulink[2], porém os resultados obtidos não foram satisfatórios segundo o autor. A fim de melhorar os parâmetros do controlador PID, o método de Ziegler-Nichols foi aplicado. Após várias rodadas de testes, foi estabelecida duas configurações dos

[1] MATLAB - https://www.mathworks.com/products/matlab.html
[2] Simulink - https://www.mathworks.com/products/simulink.html

parâmetros do controlador PID, uma para movimentações lineares e para movimentações de curva no próprio eixo. A utilização do controlador PID discreto diminui o erro causado pela diferença de diâmetros das rodas, melhorando a navegação do robô.

No artigo, Alba-Flores, Rios-Gutierrez e Jeanniton (2011), implementado um controlador PID para navegação autônoma de robôs moveis utilizando uma placa FPGA (*Field Programmable Gate Array*, em português Arranjo de Portas Programáveis em Campo). Para minimizar erros devido mau funcionamento de hardware e se concentrar na avaliação do controlador PID discreto, o sistema projetado usa uma quantidade mínima de hardware para realizar a navegação. O hardware do robô consiste em oito sensores de distância posicionados ao redor do chassi do robô, um leitor de etiquetas RFID (*Radio-Frequency Identification*, em português Identificação por Radiofrequência) e dois motores DC (*Direct Current*, em português Corrente Contínua). O robô móvel foi testado em dez tipos de diferentes modelos de ambientes internos com obstáculos, com o intuito de testar a habilidade do sistema de lidar com diferentes situações em tempo real. O objetivo dos testes era do robô atravessar o ambiente com sucesso e localizar uma etiqueta RFID. Para a modelagem do robô móvel e do controlador PID foram utilizados o MATLAB e o Simulink. O ajuste do controlador PID foi realizado utilizando o método Ziegler-Nichols para encontrar uma base dos parâmetros do controlador, e posteriormente foi aplicado o método empírico para melhorar a resposta do controlador. Os resultados mostraram que a utilização dos layouts propostos forneceram uma forma confiável de comparar o desempenho do controlador PID discreto, servindo como referência para comparar diferentes tipos controladores e configurações.

Em Liu et al. (2004), é implementado um robô aspirador com duas rodas, cada uma com seu próprio motor. Também são utilizados sensores ultrassônicos, onze no total, posicionados da extremidade esquerda até a direita do robô. Este trabalho não faz uma avaliação de desempenho do robô, mas apresenta algumas das movimentações que o robô pode realizar dentro de um ambiente e como os dados dos sensores influenciam o robô na tomada de decisões da sua movimentação.

Com base nessa contribuições o presente trabalho propõe o desenvolvimento de um robô móvel autônomo, com a definição do *hardware* e do *software* de um protótipo, selecionando um conjunto de sensores relevantes para que o robô consiga entender o ambiente em que está atuando, onde os dados coletados por esses sensores serão utilizados como entradas para o seu sistema de controle.

O sistema de controle do robô irá ditar como ele deve se comportar com os dados coletados, a fim de garantir tanto a execução da atividade como a integridade do robô. Controladores PID discretos serão utilizados em conjunto com o sistema de controle, para garantir a execução das movimentações do robô. Para que o sistema de

controle em conjunto o controlador PID discreto seja avaliado, será definido um conjunto de movimentações que o protótipo terá suporte, usando como sinal de referência o sensor de giroscópio presente no robô. Também será desenvolvida uma ferramenta de avaliação, utilizando visão computacional, para que durante a execução dos testes de movimentação seja possível acompanhar o ponto de inicio, a movimentação e ponto de parada que o robô executou.

1.2 Objetivos

O objetivo desse trabalho é desenvolver um protótipo de um robô móvel autônomo, definindo seus componentes de hardware e software, e avaliar como um controlador PID digital pode auxiliar na sua movimentação. Como objetivos específicos destacam-se:

- Desenvolver um protótipo funcional de um robô com um sistema de controle de movimentações;
- Definir um ambiente e um método de experimentação para avaliação do controlador PID discreto;
- Determinar se o uso dos componentes selecionados, em conjunto com o controlador PID discreto e o controle de movimentações, garantem uma movimentação precisa.

1.3 Organização do Trabalho

O trabalho está organizado em cinco (5) capítulos. Além deste capítulo de introdução, o Capítulo 2 demonstra a fundamentação teórica dos assuntos utilizados para o desenvolvimento deste trabalho. O Capítulo 3 descreve os componentes de hardware, arquiteturas utilizadas e detalhes da parte lógica do robô, assim como a integração com o controlador PID discreto. No Capítulo 4 explica a metodologia para avaliação de desempenho e os resultados dos experimentos. Por fim, no Capítulo 5 são feitas as considerações finais, relatos das dificuldades encontradas durante o desenvolvimento e trabalhos futuros do projeto.

2 Fundamentação Teórica

2.1 Teoria de Controle

Um sistema de controle é a combinação de componentes que agem em conjunto para atingir determinado objetivo esperado, projetados para realizar uma tarefa específica (OGATA, 2003). É composto essencialmente por um processo, uma planta a ser controlada e os distúrbios presentes no ambiente.

O processo é uma sequência de operações para atingir algum resultado, dentro de um sistema controlado é qualquer operação para qual se deseja realizar algum tipo de controle para tornar o comportamento dessas operações mais consistentes e estáveis. Dessa forma, possui uma entrada, uma saída e realiza uma determinada ação sobre uma planta, sendo este o objeto de controle em si. Esses tipos de sistemas estão sujeitos a perturbações, que tendem a afetar de maneira adversa o valor de saída do sistema de controle (OGATA, 2003).

Segundo BEGA (2006), existem três termos fundamentais que estão presentes em qualquer processo:

1. **Variável controlada:** é a variável que se deseja manter em um determinado nível, sendo a variável envolvida no processo de controle;
2. **Valor desejado:** é o valor de referência para a variável de controle se manter, também conhecido como *setpoint* (SP);
3. **Variável manipulada:** influencia diretamente a variável controlada e pode ser alterada quando necessário, levando o valor da variável controlada ao valor desejado.

Resumidamente, o sistema de controle utiliza a variável manipulada para alterar o valor da variável controlada de modo a igualar ou ficar próximo ao valor desejado dentro desse processo de controle, mesmo na presença de distúrbios no sistema.

2.2 Relação: Função de Transferência x Modelagem de um Sistema

Na teoria de controle, as funções de transferência são utilizadas para caracterizar as relações de entrada e de saída de componentes ou de sistemas que podem ser descritos por equações diferenciais lineares invariantes no tempo (OGATA, 2003).

Em outras palavras, a função de transferência é um modelo matemático que expressa o método operacional da relação entre a variável de saída à de entrada, fornecendo uma descrição das características dinâmicas, independentemente da descrição física do sistema. Se a função de transferência de um sistema for conhecida, a saída ou a resposta poderá ser estudada, visando o entendimento da natureza do sistema (OGATA, 2003). Essa função é representada matematicamente pela Equação 2.1 a seguir:

$$G(s) = \frac{Y(s)}{X(s)} \tag{2.1}$$

onde $X(s)$ é o sinal se entrada de um componente e $Y(s)$ é o sinal de saída desse mesmo componente no domínio s.

2.3 Controlador PID Clássico

Um controlador PID, é um sistema de controle com realimentação (Figura 1), onde é estabelecida uma relação de comparação entre o sinal de saída e o sinal de referência. A técnica consiste em calcular um valor de atuação sobre o processo (planta) a partir das informações do valor desejado (sinal de referência) e do valor atual da saída do processo (realimentação). Este valor de atuação sobre o processo é transformado em um sinal adequado ao atuador, devendo garantir um controle estável e preciso.

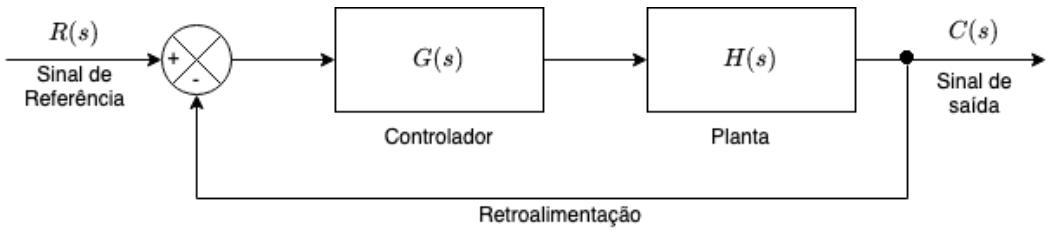


Figura 1 – Diagrama do sistema de controle com retroalimentação, Fonte: Autor.

A função de transferência desse tipo de sistema está sendo representada pela Equação 2.2.

$$C(s) = \frac{G(s)}{1 + G(s)H(s)}R(s) \tag{2.2}$$

O controle PID, como o nome sugere, é composto por três tipos de coeficientes: proporcional, integral e derivativo (BEGA, 2006). A utilidade dos controles PID está na sua aplicabilidade geral à maioria dos sistemas de controle. Em particular, quando a função de transferência da planta $H(s)$ não é conhecida, controladores PID se mostram os mais úteis (OGATA, 2003).

A maioria dos controladores PID é ajustada em campo, diferentes tipos de regras de sintonia de seus ganhos existem na literatura, como métodos de sintonia empíricos, onde exige conhecimento para definir os parâmetros de ganho dos coeficientes ou métodos que possuem uma heurística (ZIEGLER; NICHOLS et al., 1942), onde existe um processo e funções para definir. O processo de sintonia também pode ser obtido por métodos automáticos (ÅSTRÖM; HÄGGLUND, 1984).

2.3.1 Componentes de Controle do Controlador PID

Cada um dos coeficientes que compõem o sistema PID desempenha uma tarefa e um efeito diferente, fazendo com que o sinal de erro seja minimizado e o sistema controlado. Esses coeficientes estão atrelados aos parâmetros de ganho, que faz com que cada componente tenha maior influência na sua atuação dentro do sistema de controle (OGATA, 2003).

O componente proporcional produz um sinal que é proporcional à amplitude do erro, esse erro pode diminuir com o ganho, entretanto nunca é completamente zerado. Já o componente integral produz um sinal que é proporcional ao erro acumulado, ou seja, soma o termo de erro ao longo do tempo, o que implica numa reação lenta, que aumenta gradativamente ao longo do tempo conduzindo o erro de estado estacionário para zero. Por ultimo, o componente derivativo fornece uma correção antecipada do erro, fazendo com que o sinal de saída diminua se a variação do erro mudar rapidamente, melhorando a estabilidade do sistema.

Dessa forma, o controlador PID considera que um sinal de controle $U(s)$ será aplicado a uma planta $H(s)$. Este sinal de controle é gerado utilizando como entrada o sinal de erro $E(s)$ calculado pela equação $e = r - c$ de erro atuante (Figura 2).

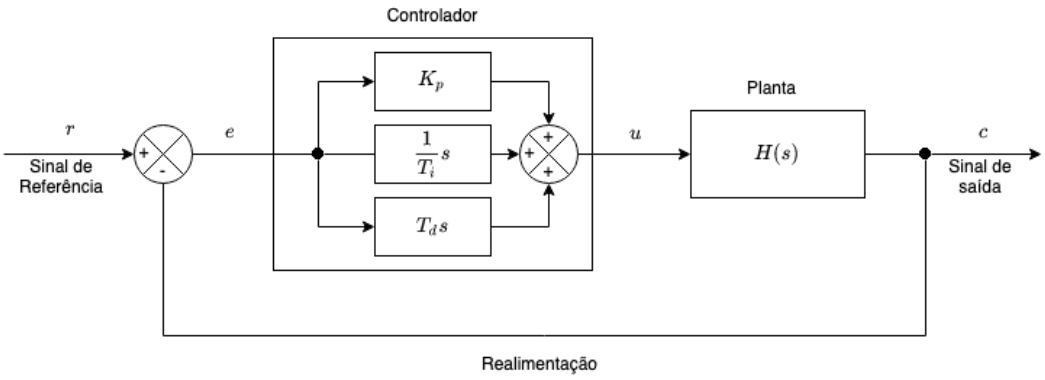


Figura 2 – Diagrama do sistema de controle PID, Fonte: Autor.

A equação tradicional do controlador PID no domínio do tempo é dada pela

Equação 2.3 mostrada a seguir:

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(t)dt + T_d \frac{de(t)}{dt} \right) \tag{2.3}$$

onde, $u(t)$ é a variável de controle e $e(t)$ é a variável de erro. A variável de controle $u(t)$ é portanto a soma de três termos: p que é proporcional ao erro, o termo i que é proporcional à integral do erro e o termo d que é proporcional à derivada do erro. Os parâmetros do controlador são proporcionais ao ganho K_p, à constante de tempo integral T_i e à constante de tempo derivativa T_d.

Segundo Ogata (2003), a função de transferência do controlador PID é dado pela Equação 2.4.

$$\frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{T_i \times s} + T_d \times s \right) \tag{2.4}$$

Os parâmetros do controlador K_p, T_i e T_d são escolhidos em função do desempenho final esperado do sistema. A determinação desses parâmetros é chamada de sintonia do controlador.

2.3.2 Controlador PID Digital

Os sistemas de controle modernos, em sua grande maioria, são implementados utilizando microcontroladores, isso se deve pela sua larga disponibilidade e seu baixo custo (OGATA, 2003). Originalmente, controladores PID são aplicados em ambiente de tempo contínuo, utilizando entradas analógicas para realizar o controle. Quando é utilizado um microcontrolador para implementar um sistema de controle, todo o processamento do sinal é feito em uma instância discreta de tempo.

Sendo assim, para implementar uma lei de controle de tempo contínuo em um sistema discreto, é necessário discretizar a derivada e a integral que aparecem na lei de controle. Em sua versão discreta, a lei de controle interpreta a ação integral como uma soma e a ação diferencial como uma diferença. O tempo contínuo se torna uma amostra de tempo em um intervalo fixo, chamado de período de amostra (ÅSTRÖM; HÄGGLUND, 1995).

Para os dados chegarem ao microcontrolador é necessário realizar uma conversão de sinal. Como representado na Figura 3, essa transformação de sinal acontece na entrada do PID, passando de um sinal analógico para um digital e na sua saída, onde o oposto ocorre. A precisão desse dado depende muito da resolução de leitura do sensor, sendo mais um dos pontos de atenção que um controlador PID discretos deve ter (ÅSTRÖM; HÄGGLUND, 1995).

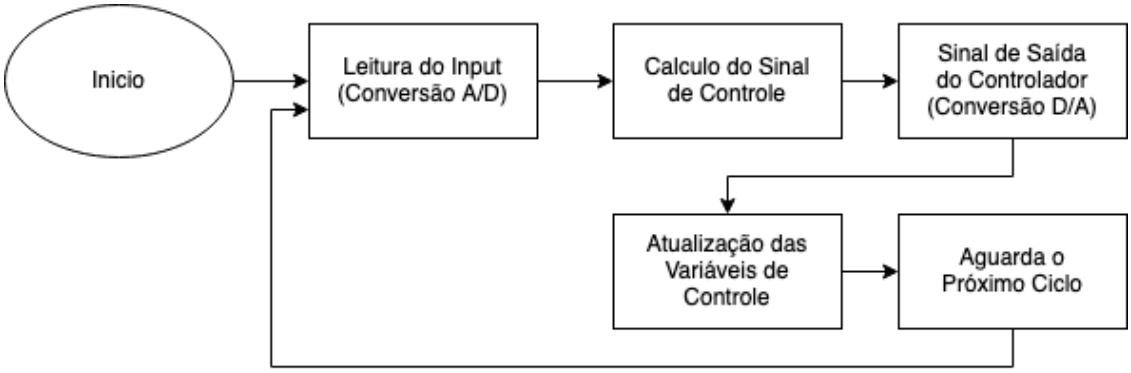


Figura 3 – Diagrama de fluxo de um controlador PID digital, Fonte: Autor.

Dessa forma, controladores PID discretos lidam com uma representação digital dos valores dos sensores amostrados dentro de um período de tempo, i.e., em tempo discreto.

2.4 Robótica: Ações Primitivas e Paradigmas

Para a execução das suas atividades o robô móvel autônomo deve ser capaz de perceber o ambiente à sua volta, tomar decisões com base nessa percepção e executar a ação que decidir com pouco erro, a fim de atingir o objetivo de sua tarefa, sendo capaz de detectar obstáculos que possam interferir no seu objetivo e saber como evitá-los. Esses desafios podem ser compreendidos a partir de três perguntas que o robô móvel precisa ter a capacidade de responder: ‘‘Onde estou?’’, ‘‘Onde vou?’’ e ‘‘Como eu vou?’’, cada uma endereçada a uma subcategoria: Localização, Mapeamento e Planejamento. (SANTOS, 2010).

A Localização é o processo por meio do qual o robô estima onde está, em relação a algum modelo do ambiente, utilizando os dados dos sensores para atualizar sua posição e guiá-lo. Se um robô não sabe sua localização, pode ser difícil determinar o que fazer a seguir. O mapeamento é como o robô percebe o ambiente, possuindo um mecanismo de detecção que possibilite a localização de obstáculos, detecção de distâncias, observação de pontos de referência, entre outros. Já o planejamento indica uma preocupação geral como o robô irá encontrar caminhos de acordo com os objetivos e diretrizes da atividade que o robô está executando.

Dessa forma, aplicar o paradigma certo torna a resolução de problemas mais fácil. Portanto, conhecer os paradigmas da robótica é crucial para ser capaz de programar com sucesso um robô para uma aplicação específica (MURPHY, 2000).

2.4.1 Ações Primitivas de um Robô

Sentir, planejar e agir são os três princípios básicos da robótica e diferentes arquiteturas de robótica definem a relação entre essas primitivas de forma diferente (MURPHY, 2000). A fim de operar qualquer atividade o robô precisa ser capaz de sentir o que está ao seu redor, o ‘‘sentir’’ é realizado por meio dos sensores. Esses dispositivos permitem o robô detectar mudanças no ambiente e enviar os dados coletados para os sistemas de controle do robô.

Uma vez que as informações do ambiente são coletadas, o robô precisa de uma estratégia para executar a sua rotina. Possuindo uma programação, onde utiliza todas as informações do ambiente (dados dos sensores) ou conhecimentos prévios (diretivas) cria um planejamento para realizar a atividade com eficiência. A fim de completar a atividade, o robô deve executar as ações de acordo com a estratégia planejada, onde o robô usa atuadores, que são dispositivos que realizam a movimentação do robô. Podendo mover-se para trás, para frente ou realizar um movimento rotativo, agindo no ambiente em que está.

Na Tabela 1 é representado como esses três tipos primitivos podem ser visualizados dentro de sistemas robóticos, em termos de entradas e saídas, durante a execução de qualquer tipo de rotina ou atividade.

Primitiva	**Entrada**	**Saída**
Sentir	Dados dos Sensores	Informações
Planejar	Informações (Sentidas ou Conhecidas)	Diretivas
Agir	Informações ou Diretivas	Comando aos Atuadores

Tabela 1 – Primitivas do robô definidas em termos de entradas e saídas.

2.4.2 Paradigmas da Robótica

A maioria dos paradigmas descritos na literatura podem ser classificados em três categorias: hierárquicos, reativos e híbridos. Esses paradigmas são definidos pela relação entre o Sentir, Planejar e Agir e como os dados detectados são processados e distribuídos para os demais componentes do sistema do robô (MURPHY, 2000).

Paradigma Hierárquico consiste em um modelo sequencial e ordenado, onde o robô coleta informações usando seus sensores, planeja a próxima ação baseado nos dados dos sensores e diretrizes da sua programação e então executa as ações (Figura 4). Assim que a ação for realizada o ciclo se repete novamente até atingir o seu objetivo. Sistemas robóticos pertencentes a este paradigma são sistemas do tipo SPA (*Sense, Plan and Act*, em português Sentir, Planejar e Agir).

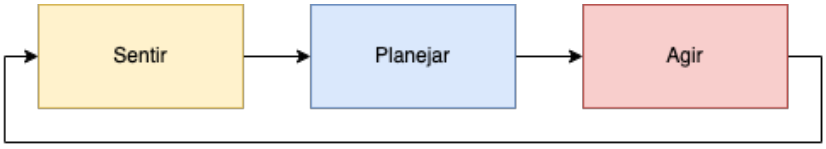


Figura 4 – Diagrama do Paradigma Hierárquico, Fonte: Autor.

Paradigma Reativo é uma técnica onde robô é capaz de tomar decisões baseadas na leitura dos seus sensores (Figura 5), unindo a percepção do robô ao ambiente com a ação. Os robôs são programados para agir utilizando uma coleção de comportamentos primitivos, para produzir resposta em tempo real com base no ambiente. Nesse método o robô possui a capacidade de se movimentar em um ambiente dinâmico e não estruturado, sendo um sistema do tipo SA (*Sense and Act*, em português Sentir e Agir).

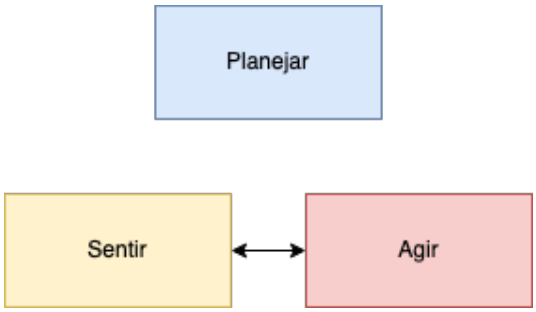


Figura 5 – Diagrama do Paradigma Reativo, Fonte: Autor.

O Paradigma Híbrido combina os paradigmas hierárquico e reativo, com o objetivo de ter a flexibilidade do robô agir diante de situações imprevistas e ao mesmo tempo planejar a sua trajetória para alcançar seu objetivo (Figura 6), buscando equilibrar planejamento com reatividade, sendo um sistema do tipo PSA (*Plan, Sense and Act*, em portugues Planejar, Sentir e Agir).

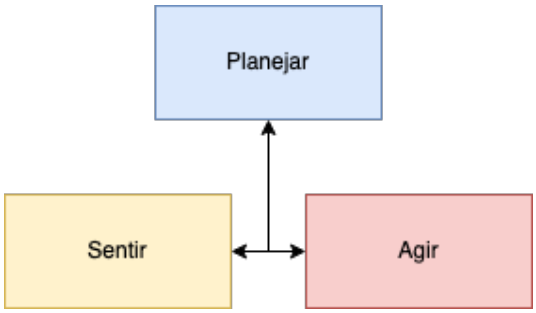


Figura 6 – Diagrama do Paradigma Híbrido, Fonte: Autor.

3 Implementação do Robô

Neste capítulo é apresentado o desenvolvimento de um protótipo de um robô móvel autônomo, especificando a estrutura do robô, suas movimentações e os componentes que serão utilizados para a construção, como chassi, sensores e atuadores. Também será especificado como foi implementado o sistema de controle do robô, apresentando o paradigma utilizado, como o protótipo tem a percepção do ambiente e como esse sistema funciona em conjunto com o controlador PID discreto.

3.1 Plataforma de Prototipagem Utilizada

O Arduino[1] é uma plataforma de prototipagem eletrônica de código aberto e hardware livre, que tem custo relativamente baixo, facilidade de uso, suporte a variados tipos de componentes e extensa documentação disponível com uma grande comunidade que desenvolve códigos para essa plataforma (EVANS; NOBLE; HOCHENBAUM, 2013), sendo uma das opções comuns em prototipação de sistemas embarcados.

As placas Arduino possuem uma variedade de modelos, cada um dispõe de quantidade diferente de periféricos que podem ser conectados. De modo geral, as placas possuem um microcontrolador, memória RAM, EEPROM e FLASH, e pinos de entrada e saída.

A linguagem de programação se assemelha bastante com a linguagem C++, porém o Arduino segue uma forma simplificada desta linguagem. Dentro das diversas placas disponíveis pela marca, escolhemos o Arduino Mega 2560 pela sua quantidade de pinos e funcionalidades se adequarem melhor no projeto proposto.

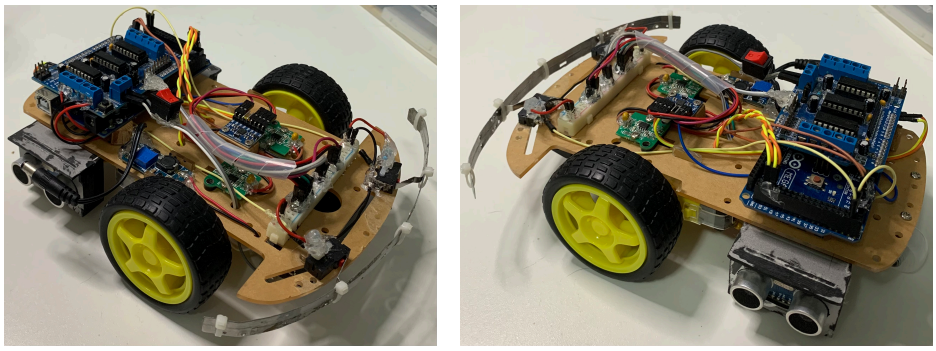
O microcontrolador utilizado no Arduino Mega 2560 é o ATMEL ATmega2560[2], um microcontrolador de 8 bits de arquitetura RISC avançada. Ele conta com 8 KB de memória RAM, 4 KB de memória EEPROM e 256 KB de memória FLASH para armazenamento do código. O Arduino também possui uma IDE que agrega uma série de ferramentas que facilitam o desenvolvimento e prototipação de sistemas embarcados, auxiliando na comunicação da placa com o computador, envio de novos programas para placa, leitura e envio de dados para placa, entre outras facilidades.

[1] Arduino - <https://www.arduino.cc/>
[2] ATMEL ATmega2560 - <https://www.microchip.com/wwwproducts/en/ATmega2560>

3.2 Robô Uniciclo

O robô tipo uniciclo é, em geral, o eleito por pesquisadores para experimentar novas estratégias de controle por possuir uma cinemática simples (BAMBINO, 2008). É uma estrutura formada por duas rodas fixas convencionais, sobre um mesmo eixo, controladas de maneira independente, e por uma roda boba que lhe confere estabilidade.

Essa configuração permite que suas velocidades angular e linear sejam controladas de forma independente, garantindo ao robô uma alta mobilidade e simples configuração de rodas (BAMBINO, 2008). Na Figura 7 temos o robô uniciclo que foi implementado no desenvolvimento desse trabalho.



(a) Dianteira do robô. (b) Traseira do robô.

Figura 7 – Robô uniciclo implementado, Fonte: Autor.

3.3 Descrição dos Componentes utilizados

O projeto do robô móvel é composto por atuadores e sensores. O microcontrolador do robô trata as informações coletadas pelos sensores e as processa de acordo com a lógica de movimentação e controle do projeto, e envia sinais aos atuadores.

O diagrama de blocos do hardware do robô é apresentado na Figura 8, nele podemos verificar a ligação entre os componentes do sistema. O robô móvel é constituído por duas (2) rodas com acionamento diferencial; motores DC com caixa de redução e eixo duplo; sensores de velocidade LM393 com discos *encoder*; módulo de controle de motor L293D *Drive* Ponte H; módulo acelerômetro e giroscópio MPU-6050; dois (2) sensores de colisão e sensores de distância ultrassônicos; e um Arduino Mega 2560. Todos os componentes são detalhados nas seções seguintes.

3.3.1 Sensores

Os sensores têm como função auxiliar o robô sobre o entendimento do ambiente para realização das suas atividades corretamente, fornecendo informações relevantes

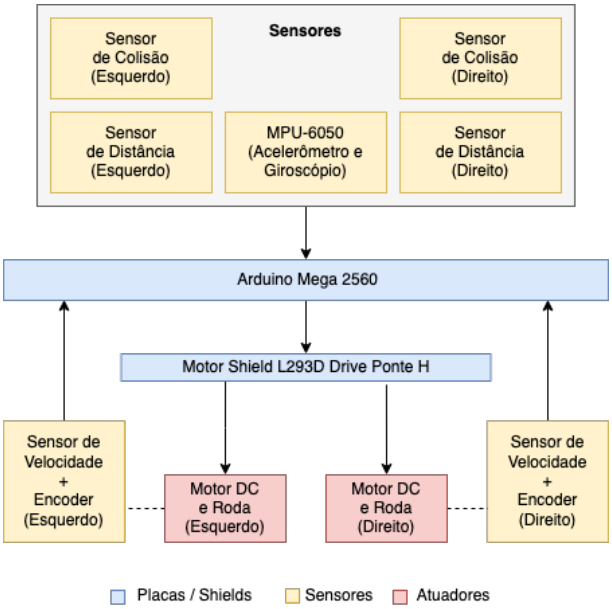


Figura 8 – Diagrama de Blocos do Hardware do Robô, Fonte: Autor.

para que perceba eventos ou mudanças de parâmetros no ambiente a sua volta. Na robótica utilizam-se principalmente dois grupos de sensores: sensores de colisão e sensores de posicionamento (BORENSTEIN; EVERETT; FENG, 1996).

Sensores de colisão são utilizados para detectar obstáculos e mapear o ambiente, podemos dividir esses sensores em duas (2) classes: sensor de contato e de distância.

O sensor de contato utilizado no robô é compostos por dois (2) botões (Figura 9) que são acionados quando o robô colide em algum obstáculo, sendo um dos tipos de sensores de colisão mais simples. Esses sensores estão posicionados na dianteira do robô e vão detectar caso o robô encontre um obstáculo a sua frente.

Figura 9 – Exemplo de sensor de contato, Fonte: https://www.addicore.com.

Para os sensores de distância, foi utilizado o módulo HC-SR04 (Figura 10). É um dispositivo para medição, bastante utilizado em robôs, sendo um sensor de baixo custo e demanda poucos recursos computacionais, nele há um circuito de controle, um transmissor e um receptor ultrassônico

Figura 10 – Módulo HC-SR04, Fonte: https://www.filipeflop.com.

Seu funcionamento se dá pela transmissão de pulsos ultrassônicos, ao atingir o obstáculo, essa onda é refletida e coletada pelo receptor. A distância é calculada pelo tempo entre a emissão e o recebimento da onda. Segundo o fabricante Elecfreaks[3], este sensor é capaz de medir distancias de 20mm a 4000mm, com precisão uma precisão de 3mm. Utilizaremos dois sensores desse tipo localizados nas laterais do robô.

Os sensores de posicionamento são utilizados para localização do robô no ambiente. No protótipo, o sensor utilizado para essa função é o modulo MPU-6050[4] (Figura 11), que possui dentro de seu circuito um acelerômetro e um giroscópio de três eixos cada, possuindo ao todo seis graus de liberdade (6DoF).

O acelerômetro é um dispositivo que mede a aceleração inercial, esta força pode ser dinâmica, como acontece na movimentação do robô, ou estática, como a força da gravidade quando o robô está parado. O Giroscópio é um dispositivo que mede a taxa de variação angular que se refere a quão rápido um objeto está girando em relação a sua velocidade e a variação do tempo do seu deslocamento. Quando utilizados de forma separada, esses sensores podem trazer leituras imprecisas sobre a posição do robô no ambiente. Uma forma de diminuir essa imprecisão é realizar a fusão dos sensores.

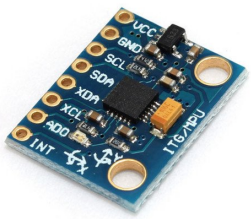


Figura 11 – Módulo MPU-6050, Fonte: https://www.filipeflop.com.

Fusão de sensores é o processo da junção dos dados de dois ou mais senso-

[3] HC-SR04 - <http://users.ece.utexas.edu/~valvano/Datasheets/HCSR04b.pdf>
[4] MPU-6050 - <https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/>

res, a fim de reduzir o erro da leitura. O principal benefício desse processo é fornecer dados mais precisos, disponibilizando dados mais confiáveis do ambiente, tornando a movimentação e comportamento esperado do robô mais preciso para a realização de suas atividades.

Para isso o MPU-6050 possui um processador de movimento digital integrado (*Digital Motion Processor* - DMP), utilizado para o processamento da fusão dos sensores isoladamente sem interferir no processamento principal do robô. Utilizaremos esse processador para realizar a fusão dos dados do acelerômetro e do giroscópio, trazendo uma leitura muito mais precisa e confiável do angulo que o robô se encontra, retornando um valor absoluto sobre a posição do robô, independente do seu estado anterior.

3.3.2 Atuadores

Os atuadores são mecanismos que o robô utiliza para interagir com o ambiente, o comportamento desses atuadores vão depender do resultado do processamento dos dados obtidos pelos sensores do robô. Neste projeto os atuadores do robô são suas rodas, sendo ativados utilizando uma interface entre eles e o Arduino. Esses tipos de interfaces são chamadas de *Shields*[5] e podem ser conectadas na parte de cima do Arduino para adicionar funcionalidades extras para a placa.

O *Motor Shield L293D Driver Ponte H* foi desenvolvido para adicionar a funcionalidade do Arduino controlar até quatro (4) motores DC, dois (2) motores servo ou dois (2) motores de passo. O *shield* permite controlar a velocidade e o sentido do giro desses motores, para isso utiliza dois circuitos integrados L293D[6] e possui internamente duas (2) Pontes H que possibilitam total controle dos motores. O shield é capaz de suportar corrente de até 600 mA por canal e até 1.2A de carga de pico, também suporta alimentação externa entre 4.5 a 25V. O *Motor Shield* pode ser observado na Figura 12.

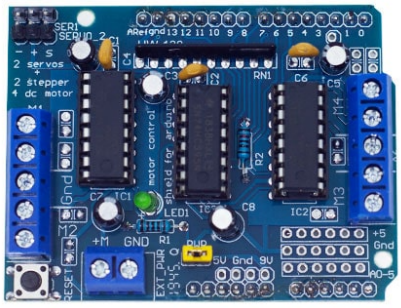


Figura 12 – Motor Shield L293D Driver Ponte H, Fonte: https://www.filipeflop.com.

[5] Arduino Shields - <https://www.arduino.cc/en/Main/arduinoShields>
[6] L293D - <https://www.ti.com/lit/ds/symlink/l293d.pdf>

A Figura 13 representa o esquema de funcionamento da ponte H. Quando as chaves $S1$ e $S4$ ficam fechadas e as chaves $S2$ e $S3$ ficam abertas, o motor rotaciona para um sentido. De forma oposta, quando as chaves $S2$ e $S3$ ficam fechadas e as chaves $S1$ e $S4$ ficam abertas, o motor rotaciona para o sentido inverso. Dessa forma, as chaves funcionam de modo alternado.

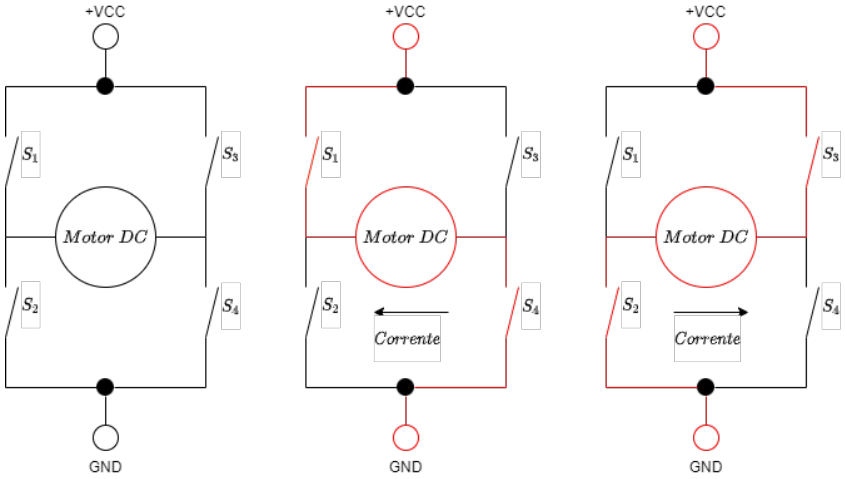


Figura 13 – Representação da Ponte H, Fonte: Autor.

3.4 Cinemática do Robô

Apesar da cinemática simples do robô uniciclo, sua movimentação não fica limitada a poucos tipos de movimentações. A Figura 14 representa todos os tipos de movimentações suportadas pelo protótipo desenvolvido neste trabalho, somando um total de seis (6) tipos diferentes. Essa variedade de movimentações é possível de ser realizada alterando as direções de movimentação dos motores do robô, onde cada roda pode girar com direções e velocidades distintas.

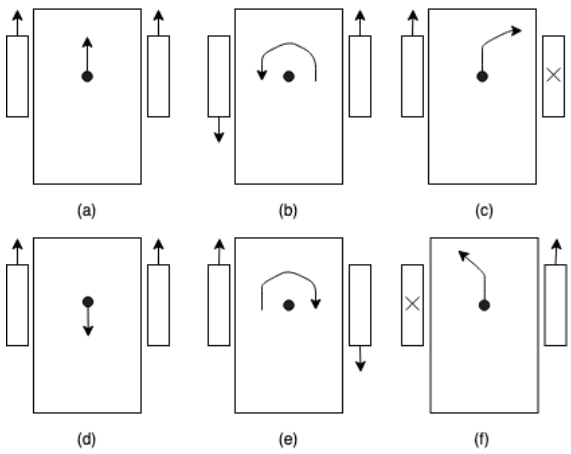


Figura 14 – Representação das movimentações do robô, Fonte: Autor.

As movimentações utilizando as duas rodas na mesma direção são representadas na Figura 14 pelas posições (a) e (d). Na posição (a) o robô possui um movimento orientado para frente, onde ambos os motores giram na mesma direção e velocidade. Se os dois motores não girarem na mesma velocidade, o robô ainda se moverá para frente, porém se o atuador direito for mais rápido que o esquerdo, o robô se moverá para frente pendendo para o lado esquerdo, o oposto ocorre se o atuador direito for mais lento que o esquerdo. Já na posição (d), se ambos os motores girarem com a mesma direção e velocidade, o robô irá para trás em linha reta. Se os dois motores não girarem na mesma velocidade, o robô se moverá, porém se o atuador direito for mais rápido que o esquerdo, o robô se moverá para a esquerda e vice-versa.

Nas posições (b) e (e), representadas na Figura 14, as duas rodas possuem movimentações opostas. Neste caso, na posição (b), o motor esquerdo possui uma movimentação direcionada para trás e o direto para frente, assim o robô girará para a esquerda no sentido anti-horário, utilizando o ponto preto como centro de rotação. Na posição (e) tem a mesma ideia, porém os motores possuem movimentações opostas da posição (b), realizando uma movimentação no sentido horário utilizando o ponto preto como centro de rotação.

Por ultimo, na posição (c) o motor direito está desativado e o motor esquerdo se move para frente, o robô realiza uma movimentação à direita usando a roda como pivô e na posição (f) o oposto ocorre. Com essa separação de movimentos, é possível separar os movimentos em três grupos, onde o primeiro grupo possui rodas girando na mesma orientação, no segundo as rodas girando em orientações opostas e na terceira somente uma das rodas está funcionando. Como cada grupo possui um deslocamento de motores distintos, cada um terá uma configuração específica para o controlador PID discreto, para a melhor execução e precisão do movimento.

3.5 Sistema de Controle de Movimentação

O software implementado no robô utiliza o paradigma reativo da robótica, em que a percepção do robô sobre o ambiente irá auxiliá-lo na tomada de decisões das movimentações que estão programadas em seu sistema.

O código implementado nesse projeto, de forma geral, inicializa os sensores e atuadores do robô e logo após realiza a execução de uma sequência de processos que está representado na Figura 15. Como é esperado o robô coleta as informações dos sensores sobre o ambiente, processa essas informações e compara com suas diretrizes e, por fim, envia os comandos para o atuador realizar as movimentações do robô.

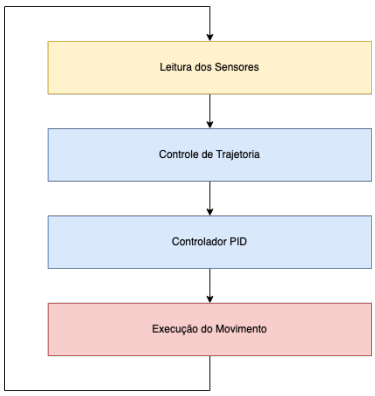


Figura 15 – Representação da arquitetura de software do robô, Fonte: Autor.

3.5.1 Controlador PID Discreto

Para a implementação de controlador PID discreto, foi utilizado uma biblioteca bastante conhecida na plataforma Arduino, a *PID Library*[7] implementada por Brett Beauregard. Nessa biblioteca é implementado um controlador PID discreto utilizando a forma incremental. Dessa forma, o controlador realiza o controle da planta calculando incrementos dos seus coeficientes de controle em relação ao tempo, utilizando os valores dos erros anteriores de cada coeficiente como inputs para a próxima interação do controlador. Segundo Åström e Hägglund (1995), a equação do controlador PID discreto utilizando a forma incremental é dada pela Equação 3.1, mostrada a seguir:

$$\Delta u(t_k) = u(t_k) - u(t_{k-1}) = \Delta P(t_k) + \Delta I(t_k) + \Delta D(t_k) \tag{3.1}$$

Na implementação de Brett, o controlador utiliza como sinal de erro $E(s)$ a equação $e = s - r$, onde s é o *setpoint* (valor-alvo) que se deseja alcançar com o controlador e r é o sinal de referência atual. Dessa forma, os incrementos do coeficiente proporcional é representado pela Equação 3.2, mostrada a seguir:

$$\Delta P(t_k) = P(t_k) - P(t_{k-1}) \approx K_p \times e \tag{3.2}$$

onde, P é o coeficiente proporcional, K_p é o parâmetros de ganho do coeficiente e e é o erro atual. Já o incremento do coeficiente integral é representado pela Equação 3.3, mostrada a seguir:

$$\Delta I(t_k) = I(t_k) - I(t_{k-1}) \approx K_i \times ei \tag{3.3}$$

onde, I é o coeficiente integral, K_i é o parâmetros de ganho do coeficiente e ei é o erro integral, sendo o erro integral anterior somado com erro atual ($ei = ei + e$). Por ultimo,

[7] Arduino PID Libray - <https://playground.arduino.cc/Code/PIDLibrary/>

o incremento do coeficiente derivativo é representado pela Equação 3.4, mostrada a seguir:

$$\Delta D(t_k) = D(t_k) - D(t_{k-1}) \approx K_d \times (e - ea) \tag{3.4}$$

onde, D é o coeficiente proporcional, K_d é o parâmetros de ganho do coeficiente, e é o erro atual e ea é o erro anterior calculado pelo controlador PID discreto.

Logo, o controlador PID discreto implementado pela biblioteca e utilizado no prototipo pode ser representado pela Equação 3.5, mostrado a seguir:

$$\Delta u(t_k) = u(t_k) - u(t_{k-1}) = K_p \times e + K_i \times ei + K_d \times (e - ea) \tag{3.5}$$

Uma vantagem dessa implementação é que como os cálculos é feito usando apenas incrementais dos coeficientes, sendo uma boa opção para ser utilizadas em microcontroladores, porém um problema com o algoritmo incremental é que ele não pode ser usado para controladores que utilizam somente os coeficientes P ou PD apenas (ÅSTRÖM; HÄGGLUND, 1995).

3.5.2 Implementação do Controle de Movimentação

O sistema de controle de movimentação do robô trabalha no intermédio entre os dados coletados do ambiente e os atuadores, operando como uma camada intermediaria, possuindo uma inteligência para tomar decisões. Essas decisões podem ser selecionadas utilizando diretrizes, comportamentos conhecidos pelo robô ou por estímulos vindo pelos dados capturados pelos sensores.

Essa dinâmica é importante para garantir tanto a execução da atividade do robô como a sua integridade. Para que o sistema consiga controlar corretamente as suas movimentações, é necessário utilizar um meio para ele acompanhe como está o andamento delas.

Dessa forma, estão sendo utilizados duas variáveis globais de controle, sendo uma variável de comando (*Command*) e outra de estado (*State*). Cada uma possui valores pré-determinados, onde o sistema de controle analisa essas variáveis e consegue determinar se uma movimentação foi iniciado, quando é o momento correto de interromper ou até iniciar uma nova.

A variável de estado é responsável por determinar o estado da movimentação, podendo assumir os valores: *MOVING*, onde indica ao sistema que uma movimentação está em execução, *CRASH*, onde notifica que o robô colidiu em algum obstáculo e *PULL*, onde notifica que uma movimentação foi finalizada e o robô está aguardando um novo comando ser disponibilizado. Já a variável de comando, é utilizada para definir

o tipo de movimentação que o protótipo deve realizar, como andar pra frente (*GO*), andar para trás (*BACK*), parar (*STOP*), virar para a esquerda (*LEFT*) ou para a direita (*RIGHT*).

Durante a execução do sistema essas variáveis de controle globais vão assumindo novos valores, dessa forma o sistema de controle de movimentação consegue se orientar em relação as suas movimentações e aos dados coletados dos sensores. Como é representado pela Figura 15, o funcionamento do sistema é composto pelas etapas: leitura dos sensores, controle de trajetória, controlador PID e execução de movimentações pelos atuadores do protótipo. Essa mesma sequencia de procedimentos pode ser representada pelo seguinte trecho de Código 3.1.

```
1 void loop() {
2   [...]
3   GetHeading();
4   UpdateSensors();
5   CrashDetect();
6   PullNexCommand();
7   ExecuteCommand();
8   [...]
9 }
```

Código 3.1 – Metódos do Sistema de Controle de Movimentações

A etapa de leitura de sensores é representado pelos métodos GetHeaing e UpdateSensores, onde são responsáveis por acionar a coleta de dados do ambiente com os sensores de colisão e de posicionamento, onde esses dados são utilizados nas etapas seguintes. Para interpretar os dados dos sensores é necessário que o protótipo tinha um entendimento do que eles significam, esse entendimento é executado pelos métodos CrashDetect e PullNextCommand, correspondendo a etapa de controle de trajetória da Figura 15.

Esse dois métodos alteram os valores das variáveis globais de controle. O CrashDetect é responsável por garantir a segurança do robô, utilizando os dados do sensor de colisão, representado pelo Código 3.2. Caso encontre uma anormalidade, a variável de estado do robô assume o valor CRASH.

```
1 void CrashDetect() {
2   if (bumper_l == 0 || bumper_r == 0) {
3     State = CRASH;
4   }
5 }
```

Código 3.2 – Método para verificação de obstáculos

O PullNextCommand é responsável por determinar quando o robô deve iniciar uma nova movimentação ou não, utilizando a variável de comando e de estado. O Código 3.3 representa um exemplo de movimentação, nele é representado como é o uso das variáveis globais. Nesse controle, o robô realiza um percurso em linha reta. Ao encontrar um obstáculo notificado pelo CrashDetect, move para trás, logo após realiza um movimento de meia volta (180 graus), e volta a andar em linha reta com a nova orientação, repetindo a lógica de movimentação.

```
1  void PullNexCommand() {
2    if (State == PULL) {
3      if (Command == NONE || Command == TURN) {
4        State = MOVING;
5        Command = GO;
6      } else if (Command == BACK) {
7        State = MOVING;
8        Command = TURN;
9
10       setTargetHeading(180.0);
11     }
12   } else if (State == CRASH) {
13     if (Command == GO) {
14       State = MOVING;
15       Command = BACK;
16     }
17   }
18 }
```

Código 3.3 – Exemplo de algoritmo de movimentação para o Sitema de Controle de Movimentos

Por ultimo, o método ExecuteCommand é representado pelas etapas restantes representadas da Figura 15, onde utiliza como entrada o valor da variável de comando para a seleção do tipo de movimentação que o robô vai realizar. Cada valor dessa variável de comando corresponde a um método de movimentação presente no robô, como demonstrado pelo Código 3.4, onde cada método ao ser chamado executa o controlador PID discreto correspondente à movimentação desejada.

```
1  void ExecuteCommand() {
2    switch(Command) {
3      case STOP:
4        StopCommand();
5        break;
```

```
6       case GO:
7         GoCommand();
8         break;
9       case BACK:
10        BackCommand();
11        break;
12      case TURN:
13        TurnCommand();
14      case NONE:
15        break;
16    }
17 }
```

Código 3.4 – Método para Execução de Movimentações do Robô

Após a atuação do movimento, a variável de estado é atualizada para o estado *PULL* dentro do próprio método de movimentação, liberando assim o robô para a próxima movimentação do definido pelo sistema de controle, repetindo o processo descrito nessa seção.

4 Experimentos e Resultados

O protótipo do robô implementado não possui em seu hardware componentes que disponibilizem o envio sem fio dos dados capturados pelos sensores e nem as tomadas de decisão do seu sistema. O Arduino disponibiliza uma porta USB (*Universal Serial Bus*, em português Porta Serial Universal) com comunicação serial, porém o robô teria que estar conectado a um cabo, o que dificultaria sua movimentação e avaliação.

Dessa forma, foi selecionado uma ferramenta para realizar essa avaliação de forma externa, a biblioteca de visão computacional OpenCV[1]. É uma biblioteca de software de código aberto amplamente utilizada em empresas, grupos de pesquisa e por órgãos governamentais (BRADSKI; KAEHLER, 2008) . Escrito nativamente em C++, porém possui implementações de sua interface em outras linguagem, como Python e Java. O código utilizado para a avaliação externa foi implementado utilizando a linguagem Python e a interface disponível do OpenCV nessa linguagem.

4.1 Estimativa da Posição do Robô em um Plano 2D

O processo de estimar a localização do robô em uma imagem capturada por uma câmera, onde um ambiente é transformado em um plano 2D, é uma etapa custosa. Sendo comum usar marcadores sintéticos para torná-la mais eficiente. Dessa forma, foi utilizado dois marcadores ArUco fixadas no chassi do robô (Figura 16). O marcador ArUco é composto por borda preta e uma matriz binária interna, onde sua borda facilita sua detecção na imagem e a matriz, que contém um identificador, permite sua identificação (GARRIDO-JURADO et al., 2014).

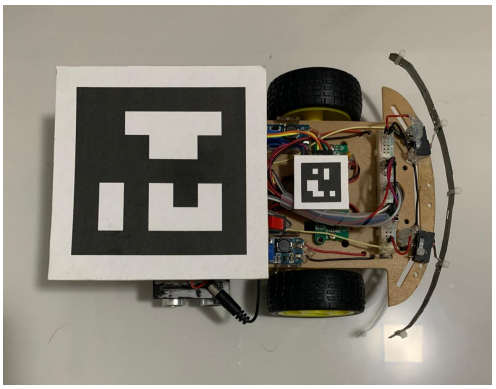


Figura 16 – Posição dos marcadores ArUco no robô, Fonte: Autor.

[1] OpenCV - <https://opencv.org/about/>

Marcadores ArUco podem ter tamanhos diferentes, para auxiliar a busca desses marcadores é utilizado um dicionário, que determina a quantidade de marcadores máxima que a imagem irá conter e o tamanho da matriz binária do marcador.

A detecção dos marcadores é feita pela análise da imagem para encontrar candidatos que podem ser marcadores, buscando artefatos com formato quadricular na imagem. Esses quadrados são segmentados afim de encontrar as bordas que o marcador ArUco possui e sua matriz interna. Após a detecção, é necessário determinar se são realmente marcadores, dividindo a imagem do marcador em células diferentes, de acordo com o tamanho do marcador e o tamanho da borda, para determinar se o marcador pertence ao dicionário que está sendo utilizado para a busca (GARRIDO-JURADO et al., 2014).

A implementação desse tipo de marcador utilizado pela biblioteca OpenCV é baseado na Biblioteca ArUco[2], desenvolvido por Garrido-Jurado et al. (2014)

4.2 Descrição dos Experimentos e os Parâmetros de Ganho dos Controladores PID

Para coleta dos experimentos foi realizada em um ambiente controlado, com condições ótimas e sem obstáculos inesperados para obstruir a movimentação do robô, a fim de avaliar o uso do controlador PID e os benefícios que ele pode trazer pra a movimentação do robô.

Dessa forma, será avaliado o sistema de controle de movimentação em conjunto com os três controladores PID discretos disponíveis. A definição dos parâmetros de ganho dos controladores foi realizada de forma empírica, em que cada parâmetro de ganho dos controladores foi ajustado levando em consideração o que cada coeficiente faz à planta que está sendo aplicada. Os valores dos ganhos utilizados estão representados na Tabela 2.

Movimento	Ganho Proporcional	Ganho Integral	Ganho Derivativo
Linha Reta	2.88	1.32	0.27
Curva (Próprio Eixo)	0.89	1.22	0.31
Curva (Roda como Pivô)	1.05	0.118	0.31

Tabela 2 – Tabela de ganhos do controle PID

Os experimentos irão avaliar se o protótipo consegue realizar as movimentações como: andar em linha reta, curvas em seu próprio eixo (90 e 180 graus) e utilizando uma de suas rodas como pivô (90 e 180 graus). Para avaliação de desempenho o robô será configurado para uma execução de um único movimento, utilizando

[2] ArUco Library - <https://www.uco.es/investiga/grupos/ava/node/26>

como sinal de referência o sensor de giroscópio. Esses experimentos serão capturados utilizando uma ferramenta implementada em Python utilizando o OpenCV como ferramenta de visão computacional, a fim de saber se ele atingiu o que foi solicitado.

Nas movimentações em curva, o robô é posicionado em um plano e uma câmera fixa fica posicionada acima dele para captura da movimentação. A câmera é responsável por registrar todo o processo de movimentação, desde o início até o momento em que o controlador PID discreto consegue posicionar o robô no *setpoint* (valor-alvo) solicitado. O cálculo do ângulo executado pela movimentação será feito utilizando os dois marcadores ArUco (Figura 16) como referência para o movimento, onde o menor será o ponto de referência para a movimentação. Na movimentação em linha reta, o robô irá percorrer uma distância de dois metros e meio (2,5m) até atingir um obstáculo fixo posicionado no final do percurso. Uma câmera é posicionada no ponto de partida, de forma que consiga registrar os pontos que marcador ArUco de maior tamanho conseguiu atingir desde o incio até o fim da movimentação.

4.3 Análise dos Experimentos

A execução dos experimentos utilizou cinco coletas de cada tipo de movimentação, em cada tipo de movimentação serão apresentados seus resultados e os cenários mais interessantes.

4.3.1 Curva no Próprio Eixo

Nesse teste vamos analisar o robô realizar um movimento de 90° e 180° em seu próprio eixo, é esperado que ele finalize sua movimentação próximo ao angulo desejado e que se mantenha em seu próprio eixo.

Na Figura 17 é apresentado um gráfico com o resultado dos cinco testes realizados na movimentação de 90°. É percebido que o robô consegue atingir um ângulo próximo do comando, porém, mesmo estando no mesmo ambiente e com mesma configuração do controlador PID discreto, o robô se comporta diferente em alguns casos, como mostra as linhas dos testes 3 e 5.

Apesar de apresentar um resultado consistente no ângulo final do movimento, com um erro médio de 2 graus, o robô não executa perfeitamente a movimentação entre seu próprio eixo. Como apresentado na Figura 18, comparando o resultado do primeiro teste com o último, podemos ver que o robô executou um movimento bastante próximo dos 90°, porém seu posicionamento difere entre os testes.

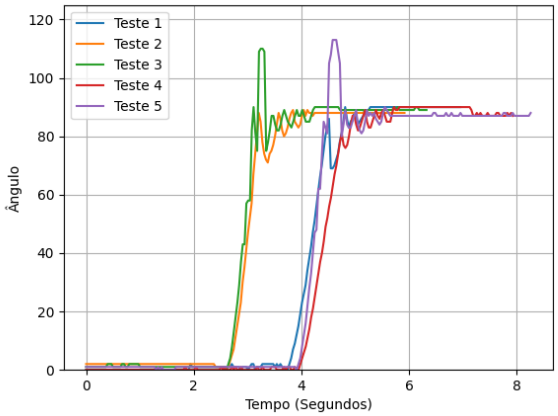
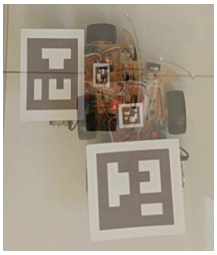


Figura 17 – Resultado dos Teste de Curva em 90°, Fonte: Autor.

(a) Teste 1.



(b) Teste 5.

Figura 18 – Relação de posição inicial e final (90°), Fonte: Autor.

Na Figura 19 é apresentado um gráfico com o resultado dos cinco testes realizados na movimentação de 180°. É percebido que o robô consegue atingir um ângulo próximo do solicitado, porém, mesmo estando no mesmo ambiente e com a mesma configuração do controlador PID, o robô se comporta diferente em alguns casos, como mostra a linha dos testes 3 e 4.

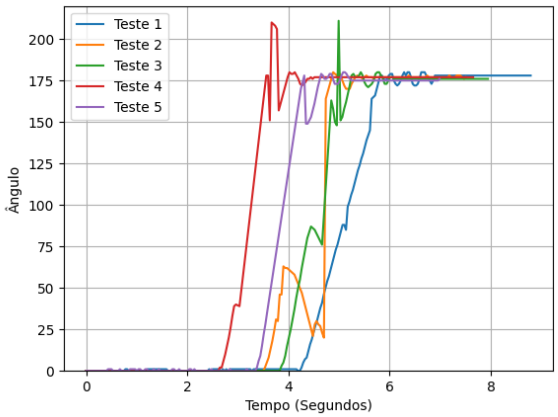


Figura 19 – Resultado dos Teste de Curva em 180°, Fonte: Autor.

Na movimentação de 180° é visto um comportamento parecido com a movimentação anterior, apresentando um resultado consistente no ângulo final do movimento, com um erro médio de 2 graus. Porém, o robô não executa perfeitamente a movimentação no seu próprio eixo. Como demonstra a Figura 20, o Teste 1 o robô não atuou um dos seus motores como deveria comparado ao Teste 2, que demonstra como essa movimentação deveria ter sido feita. Dessa forma, mesmo o robô atingindo o ângulo solicitado, o Teste 1 não atingiu o tipo de movimentação avaliado.

(a) Teste 1. (b) Teste 2.

Figura 20 – Relação de posição inicial e final (180°), Fonte: Autor.

Nas Tabelas 3 e 4, podemos ver em mais detalhes os resultados dos gráficos. É interessante notar que em todos os casos o robô conseguiu realizar a movimentação exigida, porém a execução diverge um pouco em cada caso. Em algumas momentos o robô atinge ângulos muito acima do valor desejado e em outros ele se mostra bastante estável, também é importante notar que, mesmo com um valor próximo ao valor desejado, nem sempre a execução do movimento é a ideal.

Teste	Ângulo Inicial	Maior Ângulo Registrado	Ângulo Final
1	0°	90°	90°
2	2°	89°	88°
3	1°	110°	89°
4	0°	90°	87°
5	1°	113°	88°

Tabela 3 – Tabela de Ângulos da Curva em 90°

Teste	Ângulo Inicial	Maior Ângulo Registrado	Ângulo Final
1	0°	180°	178°
2	0°	180°	177°
3	0°	211°	176°
4	0°	210°	177°
5	0°	180°	176°

Tabela 4 – Tabela de Ângulos da Curva em 180°

4.3.2 Curva com Pivô

Nesse teste vamos analisar o robô realizar um movimento de 90° e 180° utilizando uma de suas rodas como pivô, é esperado que ele finalize sua movimentação próximo ao ângulo solicitado pelo teste. Na Figura 21 são demonstrados os resultados obtidos pela movimento de 90°. Analisando o gráfico vemos que utilizando uma das rodas como pivô torna a execução do movimento mais consistente. Podemos ver que entre todos os testes a movimentação apresenta comportamentos parecidos, onde em um primeiro momento passa do ângulo exigido e estabilizando logo após esse pico.

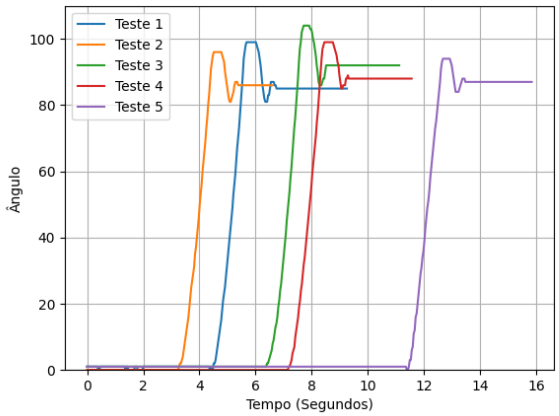


Figura 21 – Resultado dos Teste de Curva com Pivô em 90°, Fonte: Autor.

Ao contrário da movimentação realizada em seu próprio eixo, utilizar a roda como pivô mostrou que a posição final do robô é mais previsível, como mostra na Figura 22. Os Testes 2 e 3 são os que mais divergem na execução do movimento e mesmo assim tiveram um ponto de parada muito próximo.

(a) Teste 3. (b) Teste 2.

Figura 22 – Relação de posição inicial e final (90°), Fonte: Autor.

Já na movimentação de 180° graus houve uma divergência de um teste com os demais (Figura 23). No Teste 5 o robô ultrapassou o ponto de referência utilizado no controlador PID e não conseguiu se recuperar desse movimento, diferentemente dos demais que apresentaram resultados consistentes e semelhantes entre si. Na Figura 24, comparando o Teste 4 e 5, é percebido como essa falha afeta o posicionamento

final do robô. No Teste 4 o robô executa a movimentação e alcança um ângulo final próximo do exigido, porém o mesmo não ocorreu com o Teste 5.

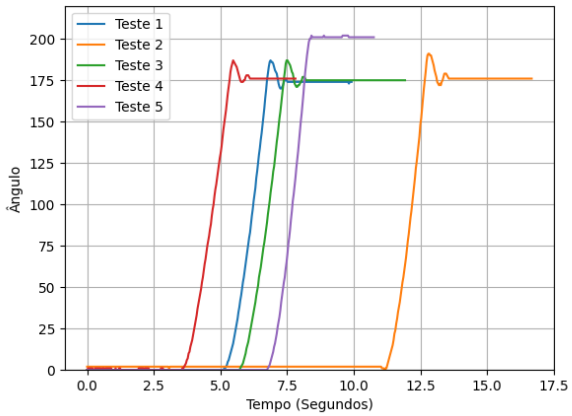


Figura 23 – Resultado dos Teste de Curva com Pivô em 180°, Fonte: Autor.

(a) Teste 4. (b) Teste 5.

Figura 24 – Relação de posição inicial e final (180°), Fonte: Autor.

Nas Tabelas 5 e 6 são apresentados os resultados principais dos experimentos dos ângulos de 90° e 180° utilizando uma das rodas como pivô. É importante citar que casos como o Teste 5 da Tabela 6 podem ocorrer com frequência. Neste caso, o robô perde a referência de sua posição em relação ao ambiente. Mesmo que esses erros possam ser pequenos e a execução de outra movimentação possa corrigir, a somatória desses eventos afeta diretamente o objetivo da atividade que o robô está realizando, fazendo com que, em alguns casos, a atividade que está exercendo não seja cumprida adequadamente.

Teste	Ângulo Inicial	Maior Ângulo Registrado	Ângulo Final
1	1°	99°	85°
2	0°	96°	86°
3	1°	104°	92°
4	0°	99°	88°
5	1°	94°	87°

Tabela 5 – Tabela de Ângulos da Curva em 90° (Pivô)

Teste	Ângulo Inicial	Maior Ângulo Registrado	Ângulo Final
1	0°	187°	174°
2	2°	191°	176°
3	0°	187°	175°
4	0°	187°	176°
5	0°	202°	201°

Tabela 6 – Tabela de Ângulos da Curva em 180° (Pivô)

4.3.3 Linha Reta

A avaliação do movimento em linha reta, o robô é posicionado em um plano e configurado para seguir o ângulo 0° do sensor de giroscópio por uma distancia de dois metros e meio.

Como apresentado na Figura 25, a movimentação em linha reta nem sempre é realizada como o esperado. Os Testes 1, 3 e 4 foram os que apresentaram melhor resultado, em que o robô conseguiu seguir o sinal de referência dado pelo giroscópio. Já os testes 2 e 5 foram os que apresentaram um resultado divergente, com o Teste 2 sendo o pior deles.

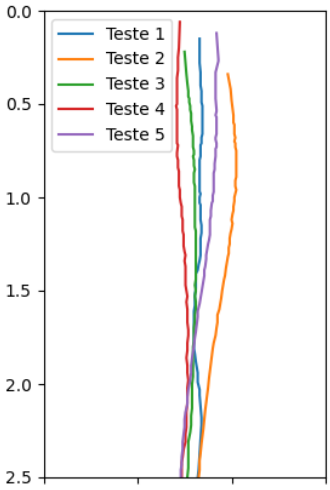


Figura 25 – Resultado dos Teste de Linha Reta, Fonte: Autor.

Esses erros podem ter sido ocasionados por algum ruído na leitura do giroscópio por um curto período de tempo, esse foi um problema visto durante a busca pelos coeficientes de ganho do controlador PID discreto. Mesmo com esses problemas o robô realizou uma movimentação satisfatória, em que conseguiu garantir a orientação do robô em linha reta.

5 Conclusões

Neste trabalho foi desenvolvido um robô uniclo com um sistema de controle de movimentações em conjunto com controlares PID discreto. Desde a definição dos componentes de *hardware*, implementação do *software* e a escolha dos coeficientes de ganho do controlador PID para cada tipo de movimentação.

De forma geral, o presente trabalho conseguiu alcançar o objetivo, os resultados mostraram que para aplicações que exigem precisão, esse tipo de hardware não oferece uma boa precisão. A atuação dos motores DC utilizados não trouxeram resultados consistentes durante a execução dos movimentos, sendo necessário utilizar outro tipo de motor com maior precisão ou sensores precisos conectados aos motores, para acompanhar melhor sua movimentação. Apesar desse problema o uso do giroscópio se mostrou eficiente para o auxilio da movimentação do robô.

O trabalho contribuiu com uma abordagem para solução de controle de movimentações em robôs, apresentando resultados satisfatórios que podem ser usadas posteriormente como solução ou base para outros problemas de controle na robótica móvel. Todos os códigos e matérias utilizados neste trabalho podem ser acessado pelo repositório do projeto[1].

5.1 Dificuldades Encontradas

Durante a implementação do robô algumas dificuldades foram encontradas, sendo elas:

- Ruido elétrico dos motores causando reinício na placa Arduino e ruídos nos sinais dos sensores, sendo necessário filtrar esse ruido com capacitores na carcaça do motor e utilizar fontes de alimentação diferentes para os motores e a placa;
- Os ângulos reportados pelo modulo MPU-6050, às vezes são inconsistentes, sendo discrepantes em relação ao real valor do ângulo em um curto período de tempo, influenciando negativamente a movimentação do robô;
- Tensão elétrica influencia bastante no funcionamento do robô, uma tensão menor faz o robô trabalhar em um condição diferente do que foi planejado;

[1] Repositório do Projeto - <https://github.com/peticormei/tcc/>

- Os atuadores utilizados no projeto não são adequados para o tipo de aplicação que exija precisão, possuindo uma precisão baixa para o tipo de movimentação e dificultando na busca dos coeficientes do controlador PID discreto.

5.2 Trabalhos Futuros

Como trabalhos futuros espera-se avaliar o uso dos outros sensores disponíveis no protótipo do robô como sinais de referência para os controladores PID discretos, afim de comparar o uso desses sinas nas movimentações e até acrescentar novas opções, i. e., andar em paralelo a uma parede utilizando os sensores de distância localizados nas laterais do robô. Também podemos utilizar esses sensores para modelar novos cenários no sistema de colisão, para que o robô consiga se antecipar a situações que não garantam sua segurança ou integridade. Outra opção seria implementar um sistema de controle que se comunique com um agente externo, adicionando a opção do robô interagir com uma central ou outros robôs, sendo capaz de enviar dados do ambiente, posicionamento atual, status da atividade ou realizar atividades compartilhada com outros robôs.

Referências

ALBA-FLORES, R.; RIOS-GUTIERREZ, F.; JEANNITON, C. Qualitative evaluation of a PID controller for autonomous mobile robot navigation implemented in an FPGA card. In: *2011 Seventh International Conference on Natural Computation*. [S.l.: s.n.], 2011. v. 3, p. 1753–1757. ISSN: 2157-9563. Citado na página 14.

ANG, K. H.; CHONG, G.; LI, Y. Pid control system analysis, design, and technology. *IEEE transactions on control systems technology*, IEEE, v. 13, n. 4, p. 559–576, 2005. Citado na página 12.

ÅSTRÖM, K. J.; HÄGGLUND, T. Automatic tuning of simple regulators with specifications on phase and amplitude margins. *Automatica*, Elsevier, v. 20, n. 5, p. 645–651, 1984. Citado na página 18.

ÅSTRÖM, K. J.; HÄGGLUND, T. *PID controllers: theory, design, and tuning*. [S.l.]: Instrument society of America Research Triangle Park, NC, 1995. v. 2. Citado 3 vezes nas páginas 19, 30 e 31.

BAMBINO, I. Una introducción a los robots móviles. 2008. Citado na página 24.

BÂRSAN, A. Position control of a mobile robot through pid controller. *Acta Universitatis Cibiniensis. Technical Series*, Sciendo, v. 71, n. 1, p. 14–20, 2019. Citado na página 13.

BEGA, E. A. *Instrumentação industrial*. [S.l.: s.n.], 2006. Citado 2 vezes nas páginas 16 e 17.

BING, L.; HUA, C. The research of the pid controller based on fpga. In: IEEE. *Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology*. [S.l.], 2011. v. 9, p. 4590–4592. Citado na página 12.

BORENSTEIN, J.; EVERETT, H. R.; FENG, L. *"Where am I?"Sensors and methods for mobile robot positioning*. 1996. Citado na página 25.

BOSSE, M.; NOURANI-VATANI, N.; ROBERTS, J. Coverage algorithms for an under-actuated car-like vehicle in an uncertain environment. In: IEEE. *Proceedings 2007 IEEE International Conference on Robotics and Automation*. [S.l.], 2007. p. 698–703. Citado na página 12.

BRADSKI, G.; KAEHLER, A. *Learning OpenCV: Computer vision with the OpenCV library*. [S.l.]: "O'Reilly Media, Inc.", 2008. Citado na página 35.

EL-TELEITY, S. A.-L. et al. Fuzzy logic control of an autonomous mobile robot. In: IEEE. *2011 16th International Conference on Methods & Models in Automation & Robotics*. [S.l.], 2011. p. 188–193. Citado na página 12.

EVANS, M.; NOBLE, J.; HOCHENBAUM, J. *Arduino em ação*. [S.l.]: Novatec Editora, 2013. Citado na página 23.

FIERROR, L. Control of a nonholonomic mobile robot: Back stepping kinematics into dynamics. *Journalof Roboticsystems*, v. 14, n. 3, p. 149, 1997. Citado na página 12.

GARRIDO-JURADO, S. et al. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognit.*, 2014. Citado 2 vezes nas páginas 35 e 36.

KUMAR, D. N. et al. Position and orientation control of a mobile robot using neural networks. In: *Computational Intelligence in Data Mining-Volume 2*. [S.l.]: Springer, 2015. p. 123–131. Citado na página 12.

LIU, Y. et al. Sensory navigation of autonomous cleaning robots. In: *Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No.04EX788)*. [S.l.: s.n.], 2004. v. 6, p. 4793–4796 Vol.6. Citado na página 14.

MAROSAN, A.; CONSTANTIN, G. Pid controller based on a gyroscope sensor for an omnidirectional mobile platform. *Proceedings in Manufacturing Systems*, University”Politehnica”of Bucharest, Machine and Manufacturing Systems …, v. 15, n. 1, p. 27–34, 2020. Citado na página 13.

MEDEIROS, A. A. A survey of control architectures for autonomous mobile robots. *Journal of the Brazilian Computer Society*, SciELO Brasil, v. 4, p. 35–43, 1998. Citado na página 12.

MENG, J. et al. Two-wheeled robot platform based on pid control. In: IEEE. *2018 5th International Conference on Information Science and Control Engineering (ICISCE)*. [S.l.], 2018. p. 1011–1014. Citado na página 13.

MURPHY, R. *Introduction to AI robotics*. Cambridge, Mass: MIT Press, 2000. (Intelligent robotics and autonomous agents). ISBN 978-0-262-13383-8. Citado 2 vezes nas páginas 20 e 21.

OGATA, K. *Engenharia de controle moderno, Ed*. [S.l.: s.n.], 2003. Citado 4 vezes nas páginas 16, 17, 18 e 19.

SANTOS, A. G. N. C. dos. *Autonomous Mobile Robot Navigation using Smartphones*. [S.l.]: November, 2010. Citado na página 20.

SANTOS, L. C. et al. Path planning for ground robots in agriculture: A short review. In: IEEE. *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. [S.l.], 2020. p. 61–66. Citado na página 12.

SELIM, G. I.; EL-AMARY, N. H.; DAHAB, D. M. A. Force signal tuning for a surgical robotic arm using pid controller. *International Journal of Computer Theory and Engineering*, IACSIT Press, v. 4, n. 2, p. 148, 2012. Citado na página 12.

WANG, K. K. et al. Research on the control system of multi-axis welding robot based on variable gain pid. In: TRANS TECH PUBL. *Advanced Materials Research*. [S.l.], 2012. v. 466, p. 784–788. Citado na página 12.

YASUTOMI, F.; YAMADA, M.; TSUKAMOTO, K. Cleaning robot control. In: IEEE. *Proceedings. 1988 IEEE International Conference on Robotics and Automation*. [S.l.], 1988. p. 1839–1841. Citado na página 12.

ZIEGLER, J. G.; NICHOLS, N. B. et al. Optimum settings for automatic controllers. *trans. ASME*, v. 64, n. 11, 1942. Citado 2 vezes nas páginas 13 e 18.