Centro de Tecnologia e Urbanismo
Departamento de Engenharia Elétrica

Evandro Junior Rodrigues

SISTEMA DE NAVEGAÇÃO AUTÔNOMA PARA PLATAFORMAS ROBÓTICAS MÓVEIS UTILIZANDO TÉCNICAS DE CONTROLE EMBARCADO

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Estadual de Londrina para obtenção do Título de Mestre em Engenharia Elétrica.

LONDRINA – PR

2014

Evandro Junior Rodrigues

SISTEMA DE NAVEGAÇÃO AUTÔNOMA PARA PLATAFORMAS ROBÓTICAS MÓVEIS UTILIZANDO TÉCNICAS DE CONTROLE EMBARCADO

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Estadual de Londrina para obtenção do Título de Mestre em Engenharia Elétrica.

Área de Concentração: Sistemas Eletrônicos
Especialidade: Sistemas de Controle

Orientador:
Prof. Dr. Leonimer Flávio de Melo

LONDRINA – PR

2014

Catalogação elaborada pela Divisão de Processos Técnicos da Biblioteca Central da Universidade Estadual de Londrina

Dados Internacionais de Catalogação-na-Publicação (CIP)

R696s Rodrigues, Evandro Junior.
Sistema de navegação autônoma para plataformas robóticas móveis utilizando técnicas de controle embarcado / Evandro Junior Rodrigues. – Londrina, 2014.
146 f. : il.

Orientador: Leonimer Flávio de Melo.
Dissertação (Mestrado em Engenharia Elétrica) – Universidade Estadual de Londrina, Centro de Tecnologia e Urbanismo, Programa de Pós-Graduação em Engenharia Elétrica, 2014.
Inclui bibliografia.

1. Robôs – Sistemas de controle – Teses. 2. Navegação de robôs móveis – Teses. 3. Robótica – Teses. 4. Robôs móveis – Teses. I. Melo, Leonimer Flávio de. II. Universidade Estadual de Londrina. Centro de Tecnologia e Urbanismo. Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

CDU 621.391

Evandro Junior Rodrigues

SISTEMA DE NAVEGAÇÃO AUTÔNOMA PARA PLATAFORMAS ROBÓTICAS MÓVEIS UTILIZANDO TÉCNICAS DE CONTROLE EMBARCADO

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Estadual de Londrina para obtenção do Título de Mestre em Engenharia Elétrica.

Área de concentração: Sistemas Eletrônicos
Especialidade: Automação e Controle de Sistemas.

BANCA EXAMINADORA

_____

Prof. Dr. Leonimer Flávio de Melo
Depto. de Engenharia Elétrica
Universidade Estadual de Londrina - UEL
Orientador

_____

Prof. Dr. Lúcio dos Reis Barbosa
Depto. de Engenharia Elétrica
Universidade Estadual de Londrina - UEL

_____

Prof. Dr. João Maurício Rosário
Depto. de Engenharia Mecânica
Universidade Estadual de Campinas - UNICAMP

05 de Dezembro de 2014

AGRADECIMENTOS

Aos meus pais, Edson e Irma, e minha irmã, Mithiele, que sempre me auxiliaram nas horas difíceis e estiveram comigo em todos os momentos da minha vida.

À minha noiva amada Elaine, pela paciência, apoio e ajuda que tem me dado durante toda esta etapa, me trazendo paz de espírito e incentivo para prosseguir.

Ao meu orientador, Leonimer Flávio de Melo, pela sua compreensão, acompanhamento e auxílio prestado ao longo do desenvolvimento deste trabalho. As lições obtidas aqui se estenderão também para futuras trajetórias, seja na pesquisa seja como profissional.

Aos meus colegas de mestrado da UEL pela amizade e convivência durante o curso.

À Universidade Estadual de Londrina e a todos os membros por oportunizar orientações e ensinamentos para o meu crescimento pessoal e desenvolvimento deste trabalho.

À CAPES pelo auxílio financeiro.

RESUMO

O desenvolvimento de sistemas de controle para robôs móveis autônomos tem se mostrado um grande desafio para a os pesquisadores até os dias atuais. Diferentes abordagens para o projeto de sistema de controle para robôs móveis autônomos vem sendo utilizadas em diversas áreas de pesquisa. Por muitos anos os pesquisadores tem construído sistemas de controle que apresentam um comportamento inteligente em ambientes controlados, com situações ideais, mas que normalmente não mantém o mesmo desempenho no mundo real. Nesse contexto foi desenvolvido um sistema utilizando técnicas de hardware in-the-Loop para operação de um sistema robótico controlado através do MatLab no desenvolvimento de um modelo cinemático e dinâmico. Desta forma o objetivo deste trabalho é apresentar um sistema de controle robusto para robôs móveis autônomos que seja capaz de operar e de se adaptar a diferentes ambientes e condições. Um sistema robótico móvel será utilizado para validação dos testes dos sistemas de controle e navegação autônoma.

ABSTRACT

The development of control systems for autonomous mobile robots has been shown as a great challenge for researchers until nowadays. Different approaches for the autonomous mobile robots control system project have been used in several research areas. For many years researchers have built control systems that show an intelligent behavior in controlled environments, with ideal situations, but that usually don't keep the same performance in real world. In this context a control system was developed in open and close mesh for the operation of a robotic system controlled by MatLab in the development of a cinematic and dynamic model. Thus, the objective of this work is to present a robust control system for autonomous mobile robots that is able to operate and adapt to different environments and conditions. A mobile robotic system will be used for validation of the tests of control and autonomous navigation systems.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DSP	Digital Signal Processor
USB	Universal Serial Bus
DC	Corrente Contínua
MA	Malha Aberta
MF	Malha Fechada
CPU	Central Processing Unit
PC	Personal Computer
ROM	Read Only Memory
RAM	Random Acess Memory
EEPROM	Electrically Erasable Programmable Ready Only Memory
ASIC	Application Specific Integrated Circuits
FPGA	Field Programmable Gate Arrays
ADC	Conversor Analógico Digital

CONVENÇÕES E LISTAS DE SÍMBOLOS

Na registro das fórmulas, as seguintes convenções foram utilizadas:

- Ra Resistência da armadura
- La Indutância da armadura
- Kt Constante de torque
- Kb Constante da FCEM
- Jt Momento de inércia
- b Atrito viscoso
- Wmax Velocidade angular máxima
- K_{RED} Relação de transmissão
- Jl Momento de inércia da carga
- bl Atrito viscoso da carga
- A,B,C,D Matrizes do sistema

1 INTRODUÇÃO

O desenvolvimento de sistemas de controle para robôs móveis autônomos tem se mostrado um grande desafio para a os pesquisadores até os dias atuais. Diferentes abordagens para o projeto de sistema de controle para robôs móveis autônomos vem sendo utilizadas em diversas áreas de pesquisa. Por muitos anos os pesquisadores tem construído sistemas de controle que apresentam um comportamento inteligente em ambientes controlados, com situações ideais, mas que normalmente não mantém o mesmo desempenho no mundo real. Existem inúmeros sistemas de controle para serem utilizados no mundo real, mas geralmente estes sistemas são limitados e não apresentam um comportamento autônomo ou inteligente.

Este trabalho tem por finalidade desenvolver um protótipo de um robô autônomo, com sistema micro controlado utilizando-se como ferramenta de desenvolvimento o MatLab totalmente integrado com o microcontrolador atmega 2560, sendo possível além de efetuar as simulações, também as implementações, conseguindo medir os resultados e compara-los para obter a confiabilidade do sistema. O objetivo principal é que de acordo com as simulações efetuadas das trajetórias geradas no MatLab sejam realizadas igualmente pela robô na prática.

Desta forma este trabalho foi dividido em 3 etapas conforme Figura 1.1. Na primeira etapa, foram desenvolvidos um modelo cinemático e um modelo dinâmico. Para a modelagem cinemática, fez-se o uso de um equacionamento já elaborado para determinar a posição do robô e desta forma conseguir simular a trajetória do robô. Para a modelagem dinâmica, desenvolveu-se o sistema de controle com a elaboração da equação de estado do sistema, verificando-se em seguida a estabilidade do sistema para a equação encontrada.

Para a segunda etapa, utilizou-se o modelo cinemático para simular uma trajetória reta de comprimento 1 metro, além de uma trajetória circular de raio igual a 1 metro e uma meia circunferência de raio igual a 1 metro. Do mesmo modo fez-se a simulação do modelo dinâmico verificando a estabilidade para o equacionamento matemático obtido.

Na terceira etapa, foi aplicado a digitalização do modelo dinâmico e a construção do simulador virtual para em seguida ser feita a implementação, onde foi aplicado os resultados teóricos na prática. Este simulador reuniu o trabalho feito anteriormente para o modelo cinemático e dinâmico. Assim, como resultado final, o robô fica apto a percorrer a trajetória criada, conseguindo cursar o mesmo caminho da trajetória simulada anteriormente.

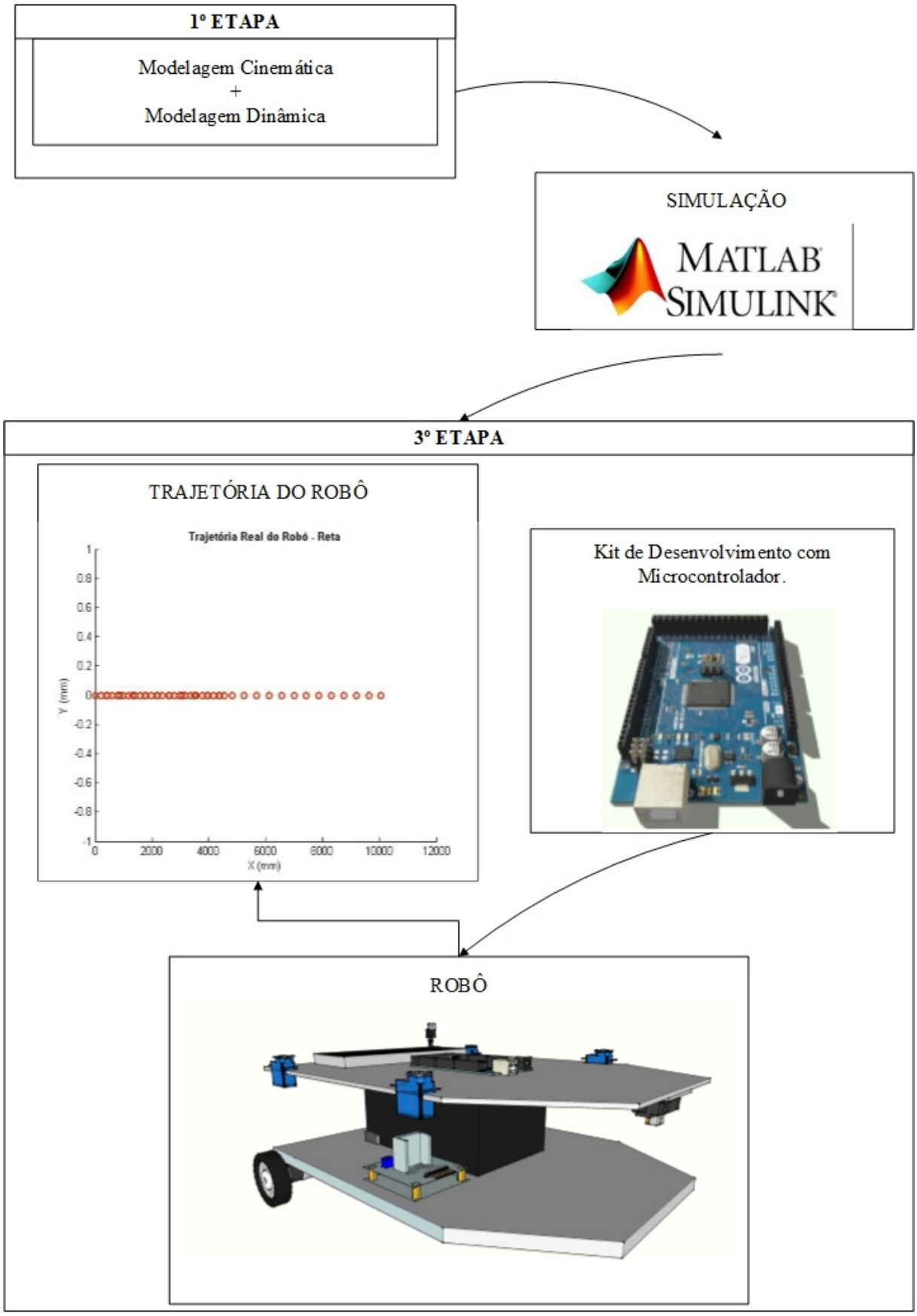


Figura 1.1: Etapas do desenvolvimento deste trabalho.

1.1 Objetivos Gerais e Específicos

O Objetivo deste trabalho é realizar a implementação em microcontrolador do controle PI embarcado para os motores de corrente contínua do robô através do ambiente MatLab para a realização de análise em tempo real da operação dos motores junto com a trajetória percorrida pelo robô.

Assim, o trabalho desenvolveu-se em 3 etapas:

1. Desenvolvimento do modelo cinemático e dinâmico
2. Simulação do modelo cinemático e dinâmico no ambiente MatLab/Simulink
3. Implementação do modelo cinemático e dinâmico no sistema de controle embarcado, formado por um kit de desenvolvimento que utiliza um microcontrolador ATmega2560
4. Aquisição de dados para comparação do sistema simulado com o sistema implementado.

1.2 Justificativas

A Robótica, em diferentes formas, permeia o plano das ideias da humanidade desde o momento em que se percebeu que o ser humano é hábil para construir coisas. É sabido também que os robôs são elementos muito poderosos da indústria, aptos a desenvolver diversas tarefas e operações de forma precisa e prescindindo dos elementos comuns de segurança e de conforto que o seres humanos carecem para a execução das mesmas tarefas (NIKU, 2013).

Assim, com o intuito de produzir um robô controlável e viável economicamente, este trabalho criou um sistema real, desenvolvido na teoria e testado na prática de um modelo cinemático e dinâmico. A partir deste modelo é possível não somente se limitar ao uso do controle de um robô, pode-se ir além e entender a teoria desenvolvida para controle de outros equipamentos, como uma cadeira de rodas.

1.3 Metodologias

Para o desenvolvimento deste trabalho, utilizou-se o ambiente de trabalho MatLab/Simulink e, através da criação de script pelo MatLab, elaborou-se um modelo cinemático e, junto com o Simulink, um modelo dinâmico. Desta forma, os resultados desta

simulação foram implementados no kit de desenvolvimento que utliza o microcontrolador ATmega 2560.

1.4 Estrutura do Trabalho

Este trabalho divide-se em 8 capítulos, sendo que o Capítulo 1 apresenta o escopo da dissertação, os objetivos gerais e específicos, justificativas e metodologia. O Capitulo 2 exibe o conceito de eletrônica embarcada. O Capitulo 3, discorre sobre os conceitos de controle em malha aberta (sistema que não utiliza o feedback do encoder) e fechada que serão utilizados neste trabalho além dos componentes de hardware que serão usados neste trabalho. O Capitulo 4 aborda os conceitos de modelagem cinemática e a modelagem dinâmica do robô. De acordo com a teoria e o equacionamento dinâmico já apresentados nos capítulos anteriores, demonstra-se a simulação de todos os assuntos abordados no Capítulo 5 para, no Capítulo 6, implementar no robô. A partir do desenvolvimento efetuado nos dois capítulos anteriores, o Capítulo 7 compara os valores simulados aos valores implementados, determinando seu erro. Após os processos listados ao longo do trabalho, o Capítulo 8 expõe a análise dos resultados e a conclusão final do trabalho.

2 SISTEMAS DE CONTROLE EMBARCADO

Este capítulo é referente a revisão bibliográfica e revisão dos conceitos de controle de malha aberta e fechada para o sistema que será desenvolvido. Também neste capítulo descrevem-se os componentes de hardware utilizados neste projeto.

De acordo com (Dorf, 2001), o objetivo permanente dos sistemas de controle é prover ampla flexibilidade e níveis elevado de autonomia. Para este trabalho o objetivo é que, uma vez programado o robô conforme o sistema de controle utilizado, torne-se desnecessária a intervenção humana para executar uma determinada tarefa.

De acordo com (Groover, 2011), os controles de um sistema automatizado podem ser tanto de malha fechada como de malha aberta.

2.1 Estado da Arte

Pesquisas sobre Robótica e desenvolvimento de tecnologia desenvolvem-se há quase meio século. A definição oficial do robô pelo Instituto da América (RIA) (SPONG; HUTCHINSON; VIDYASAGAR, 2006) foi:

"Um robô é um manipulador multifuncional reprogramável projetado para mover materiais, peças, ferramentas ou dispositivos especializados através de variáveis programadas de movimentos para uma performance ou uma variedade de tarefas. "

Nos dias de hoje há robôs utilizados para diversos tipos de atividades, tais como tarefas em indústrias, em residências e até no espaço (BEKEY; YUH, 2008). O objetivo de muitos destes trabalhos consiste em substituir a atividade humana por tarefas programadas para serem executadas por robôs em ambientes que apresentem potencial de risco, especialmente à incolumidade física, como ambientes sujeitos a explosão, radioativos, entre outros (SPONG, HUTCHINSON;VIDYASAGAR, 2006).

O ponto fulcral para a definição do robô é a reprogramação, que dá ao robô sua característica de adaptabilidade. Esta definição é aplicada a robôs manipuladores, que possuem estrutura mecânica formada por uma série de hastes conectadas por meio de juntas. A fim de desenvolver uma tarefa, requer a movimentação de suas juntas, para que sua extremidade se movimente ao longo de uma trajetória definida, de modo que sua ferramenta possa alcançar os pontos necessários, com a orientação desejada, para realização da tarefa

especificada (MARTINS, 2009). Atualmente os robôs manipuladores cumprem tarefas de deslocamento de objetos na indústria de manufatura, pintura e soldagem na indústria automobilística, além de outras (BEKEY; YUH, 2008).

Dentre as classes de robôs, ampliou-se a visibilidade e a importância, devido ao número de aplicações, dos robôs de serviço. Essa classe abrange robôs que desenvolvem serviços de utilidade aos seres humanos ou equipamentos, apartando as operações de manufatura. Inserem-se neste contexto os robôs que auxiliam em tarefas como busca e resgate, assistência doméstica (como aspirador de pó e cortadores de grama), entretenimento (esportes praticados por robôs, robôs que se comportam com animais de estimação) e assistência a pessoas com deficiência (como cadeiras de rodas robóticas e dispositivos de auxílio ao caminhar) (ROMANO, 2002).

Robôs de serviço tornou-se um grande atrativo do mercado, em 2012 registrou-se a venda de cerca de 3 milhões de robôs de uso doméstico e pessoal em todo o mundo, segundo a federação internacional de robótica, o que representou um faturamento de U$$ 1,2 bilhão. Para o período de 2013 a 2016, a previsão é de 15,5 milhões de unidades, com vendas de U$$ 5,6 bilhões (CRUZ, 2014)

Muitos robôs de serviço são robôs móveis, capazes de movimentações autônomas, equipados com atuadores e controlados por um computador embarcado (CANUDAS de WIT; SICILIANO; BASTIN, 1997).

Para o controle de robôs de serviço, é necessário verificar o aspecto construtivo do robô. Neste trabalho o robô utilizado é um robô uniciclo que, acordo com (MARTINS, 2009), possui duas rodas de tração independentes, fixas e dispostas sobre um mesmo eixo. Também dispõe de uma roda independente, que gira livremente.

Usualmente, a teoria de controle não-linear tem sido aplicada no estudo do robô móvel (KANAYAMA et al., 1990; CANUDAS de WIT C.;SORDALEN, 1992). Vários estudos abordam o projeto de controladores de seguimento de trajetória e o problema de geração de trajetórias para estes robôs (FRAGA; SOUSA; PEREIRA, 2003). Alguns controladores foram projetados com base em seu modelo cinemático, como em (CARELLI; SECCHI; MUT. 1999), entretanto, para execução de tarefas que requerem deslocamento em altas velocidades ou transporte de cargas, a consideração da dinâmica dos robôs se torna essencial (FIERRO; LEWIS, 1997; FIERRO; DAS, 2002). Em (KIM; SHIN; LEE, 2000) foi proposto um

controlador adaptativo para robôs móveis, dividido em duas partes: a primeira, baseada na cinemática do veículo, gera sinais de referência para a segunda, que compensa sua dinâmica e gera sinais de torque para comandar o robô.

De outro norte, em (De La CRUZ, 2006; De La CRUZ; CARELLI, 2006) foi sugerido um modelo dinâmico que aceita sinais de referência de velocidades lineares e angulares como entradas, apresentando um projeto de um controlador dinâmico para seguimento de trajetória baseado em tal modelo. Desponta como vantagem do controlador ali apresentado a geração de sinais de velocidade linear e angular como comandos, já que robôs comerciais usualmente possuem controladores internos e aceitam sinais de referência de velocidade, e não de torque ou tensão para seus motores (MARTINS, 2009).

2.2 Sistema em Malha Aberta

Este é o caso mais simples, onde os controles em um sistema operam sem medir a variável de saída, não havendo comparação entre o valor de saída e o parâmetro de entrada desejado ou seja, não há realimentação, conforme a Figura 2.1.

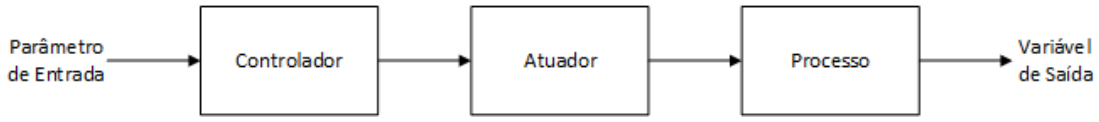


Figura 2.1: Sistema de controle de malha aberta.

No caso do robô, este sistema não apresenda o feedback do encoder, num sistema deste tipo seria necessário ter um sistema exato do seu atuador no caso o motor CC sobre a variável de processo, no caso o encoder de cada motor. Esta malha representa o sistema em relação a 1 roda do robô, ou seja, é um sistema de controle sobre cada motor. Num sistema em malha aberta, existe sempre o risco de os resultados não corresponderem ao que foi esperado por não haver o sistema de comparação do sinal de saída com o sinal de entrada e isto faz com que este sistema tenha uma desvantagem em relação ao sistema em malha fechada. Para conseguir medir qual o melhor sistema utilizar em um trabalho como este, deve-se considerar os seguinte caso:

Será utilizado um sistema de controle em Malha Aberta se:

- As ações executadas pelo sistema de controle são bastante confiável

- A função do atuador é bastante confiável
- Quaisquer forças de reação opostas às do atuador são pequenas demais para causar algum efeito sobre a atuação.

Desta forma, se nenhum destes itens forem aplicados, pode-se utilizar no robô o sistema em malha aberta.

Em relação ao robô, e que pese haver mesma motor com idênticas características para cada roda, o comportamento delas na prática não é totalmente igual, havendo uma divergência entre cada motor, o que não oferece confiabilidade para utilização em malha aberta.

2.3 Sistema do controle em Malha Fechada

Um sistema de controle de malha fechada, também conhecido como sistema de controle por realimentação, é aquele no qual a variável de saída se compara a um parâmetro de entrada e qualquer diferença entre eles é utilizada para fazer com que a saída esteja em conformidade com a entrada. O sistema de controle em malha fechada é formado por seis elementos básicos, conforme listado abaixo e demonstrado na Figura 2.2

1- Parâmetro de entrada;

2- Processo;

3- Variável de saída;

4- Sensor por realimentação;

5- Controlador;

6- Atuador.

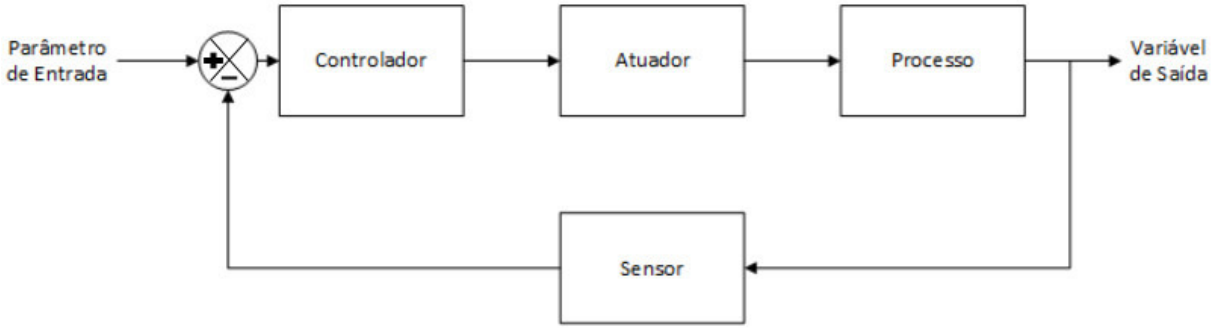


Figura 2.2: Sistema de controle de malha fechada.

O Parâmetro de entrada, nesse caso, é o valor que deseja na saída. Desta forma, o robô objeto deste trabalho utiliza um encoder para quantizar a variável de saída e fechar a malha entre a entrada e a saída. Assim, o sistema utiliza o feedback do encoder. Já o controlador compara a saída com a entrada fazendo ajustes que são necessários no processo para reduzir as diferenças entre elas. Os ajustes feitos pelo controlador são executados pelo atuador.

Neste trabalho o controlador utilizado é um microprocessador ATmega 2560 para fazer o controle dos dois motores CC. Segundo (Groover. 2011), com o controlador digital por computador, é possível executar algoritmos de controle mais complexos que os executados nos modos de controle proporcional-integral-derivativos convencionais usados por controladores analógicos.

No caso do Robô, na saída existem dois sensores, que são um encoder para cada motor. O sinal de saída do encoder está conectado à entrada do microprocessador, assim o microprocessador processa estas informações e as envia para suas portas de saída que vão para um conversor Digital Analógico(DAC), fazendo a conversão do sinal digital processado para um sinal analógico. Este sinal serve como sinal de entrada para a ponte H do L298, uma vez que somente as saídas do microcontrolador não é suficiente para controlar o motor, o L298 faz com que o motor opere de acordo com o controle programado. Ainda, vale ressaltar que o microcontrolador ATmega 2560 já contém os conversores ADC utilizados na Figura 2.3.

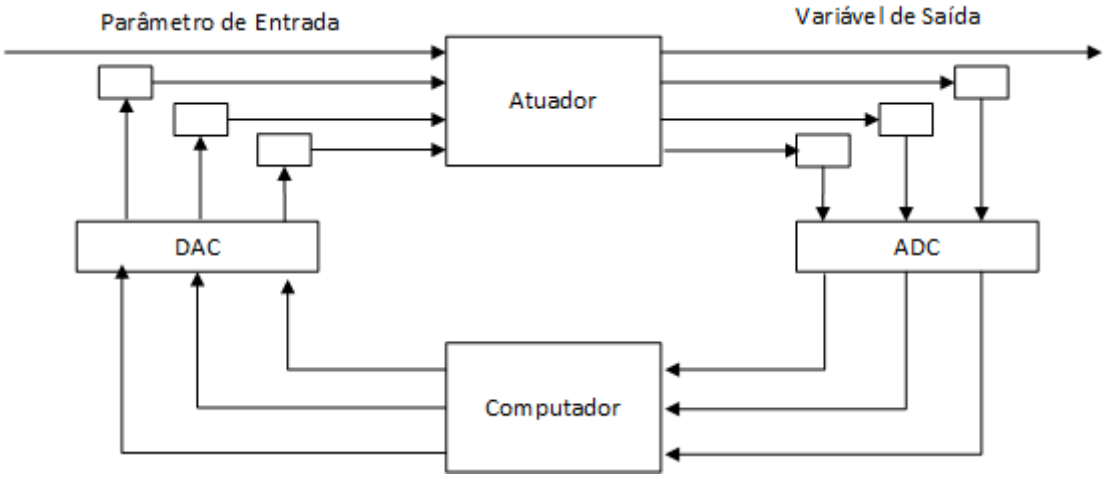


Figura 2.3: Controle digital que será utilizado.

A Figura 2.3 ilustra o controle digital no microcontrolador, já com os conversores AD e DA embutidos no próprio microcontrolador.

2.4 Controle do Robô

Em um sistema onde existe um controle independente de rodas, mesmo utilizando um motor com características similares para cada roda, quando é aplicada a mesma potência de cada uma das rodas, normalmente não resulta, para o veículo, percurso em linha reta.

Tal fenômeno decorre das diferenças da parte construtiva de cada motor, assim como pela diferença de superfície, além de outros fatores, como a fixação da roda em cada eixo do motor. Desta forma, a melhor solução para o controle deste projeto é a utilização de um controlador de circuito em malha fechada que significa o uso do feedback do encoder, o qual ajusta a potência aplicada a cada um dos motores com base na diferença das taxas de rotação.

Para começar o projeto de controle do robô, o primeiro objetivo é controlar com o máximo possível de exatidão a velocidade de cada um dos motores DC que serão embutidos em cada roda do robô. Assim, de acordo com (Dorf, 2001), os passos que deverão ser perseguidos para o controle podem ser descritos da forma subsequente:

1º - Finalidade do Sistema. O objetivo deste trabalho é o controle da velocidade com exatidão dos motores DC.

2º - Identificar as variáveis que se deseja controlar. Neste caso é a velocidade dos motores DC.

3º - Inclusão de Sensores para medir a variável controlada para tentar atingir com tamanha exatidão resultado esperado.

A primeira tentativa é fazer uma configuração de um sistema que conduza ao desempenho de controle desejado. Esta configuração consistirá de um sensor de um processo sob controle (sendo o processo o robô), de um atuador e de um controlador. Identificada a malha de controle, devemos agora escolher equipamentos controláveis, que no caso do robô são os motores DC.

Importante mencionar que a escolha do atuador é considerada uma etapa relevante para o desenvolvimento do projeto, que requer certo cuidado para que no processo de controle consiga ajustar efetivamente o desempenho do processo.

Assim, neste trabalho o dispositivo de controle utilizado é o microcontrolador ATmega 2560, O Atuador é o motor DC, o processo é o Robô, e o Sensor é o encoder conforme a Figura 2.4.

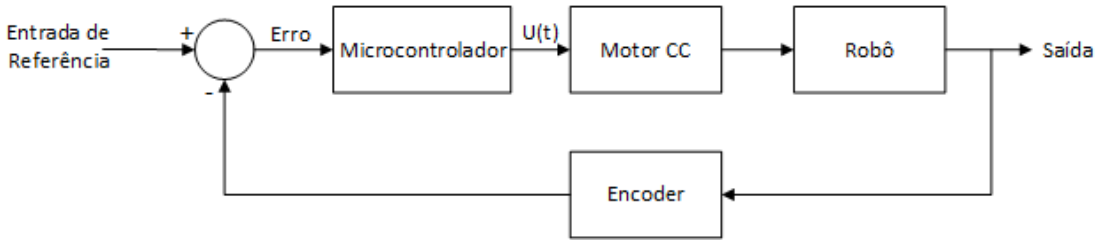


Figura 2.4: Controlador do circuito em malha fechada que será utilizado.

Na Figura 2.4 acima a Entrada de Referência é o valor que eu desejo obter na saída, desta forma o Erro é o valor da saída menos o sinal de entrada onde é possível obter o sinal da saída pelo encoder apresentado na figura acima. Seguindo o desenvolvimento da Figura 2.4, o microcontrolador é quem vai processar o sinal de entrada e controlar o motor, ao passo que o U(t) é o valor da entrada do motor cc já amplificada pela L298 que será suficiente para fazer o motor girar na velocidade desejada. O bloco do robô simboliza sua parte dinâmica, conforme o peso e atrito com o solo que fará com que vá na direção desejada com os controles que serão ajustados nos próximos capítulos deste trabalho.

Nesta etapa, identificada a malha de controle já ajustada para o caso real, será feita a seleção de um controlador. Conforme (Dorf, 2001) explica, um controlador quase sempre consiste de um amplificador subtrator que irá comparar a resposta desejada com a resposta real e em seguida encaminhar este sinal de medida de erro a um compensador.

Finalmente, a última tarefa no processo de controle do robô é o ajuste dos parâmetros do sistema, a fim de obter o desempenho desejado. A Figura 2.5 retrata o procedimento que foi utilizado para o projeto do sistema de controle.

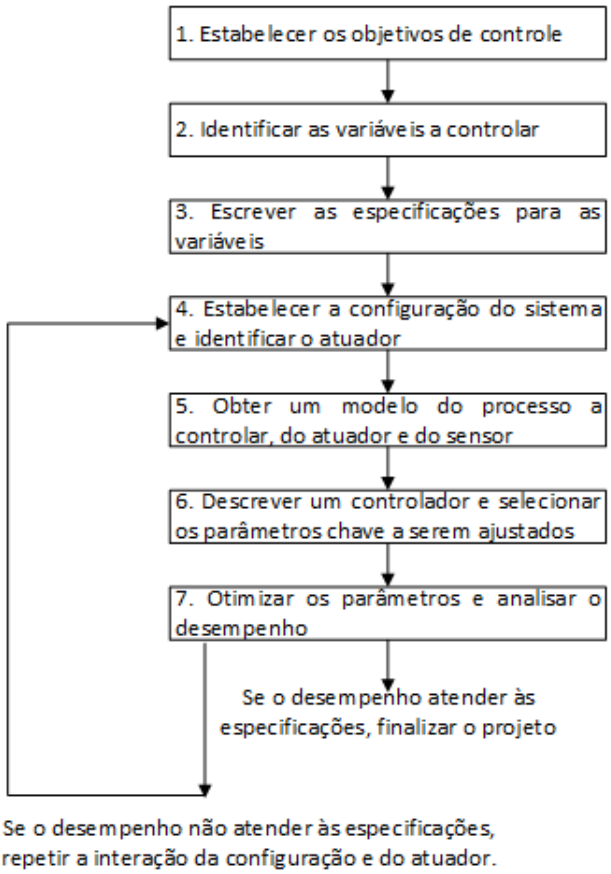


Figura 2.5: Mapa do ajuste dos parâmetro do controle do robô.

Antes de implementar o controle em malha aberta e fechada, é necessário levantar as características de cada componente utilizado.

2.5 Componentes Utilizados no Desenvolvimento do Projeto

Na Figura 2.6 é apresentado o desenho em 3D do robô que será utilizado neste trabalho, junto com seus acessórios.

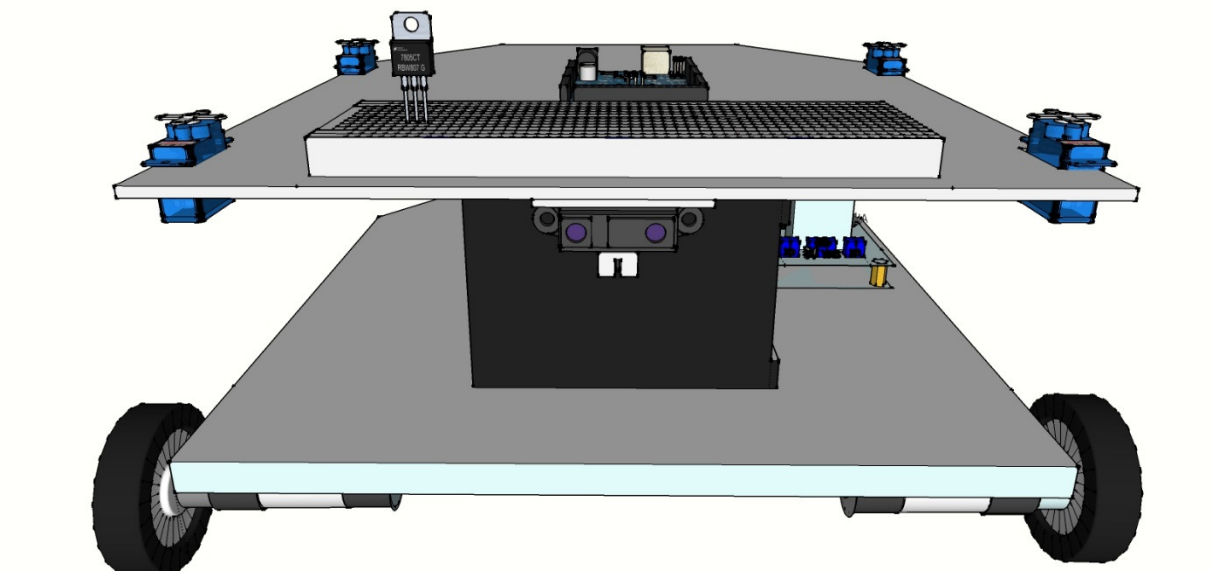


Figura 2.6: Desenho 3D do Robô.

De acordo com (Groover, 2011), para implementar a automação e o controle de processos, o computador de controle deve coletar dados do processo de produção e transmitir sinais a ele. O computador digital opera com dados digitais, enquanto alguns dados do processo são contínuos ou analógicos. Nesse sentido, deve-se comandar estas diferenças na interface entre o processo e o computador. No caso do robô, os componentes que serão utilizados para a implementação da interface são os seguintes:

1 – Sensores para quantizar as variáveis contínuas e discretas do processo;

2 – Atuadores que acionam os parâmetros contínuos ;

3 – Dispositivos que convertem sinais analógicos contínuos em dados digitais;

4 - Dispositivos que convertam dados digitais em sinais analógicos;

5 – Dispositivos de entrada/saída para dados discretos.

A Figura 2.7 ilustra os itens numerados acima.

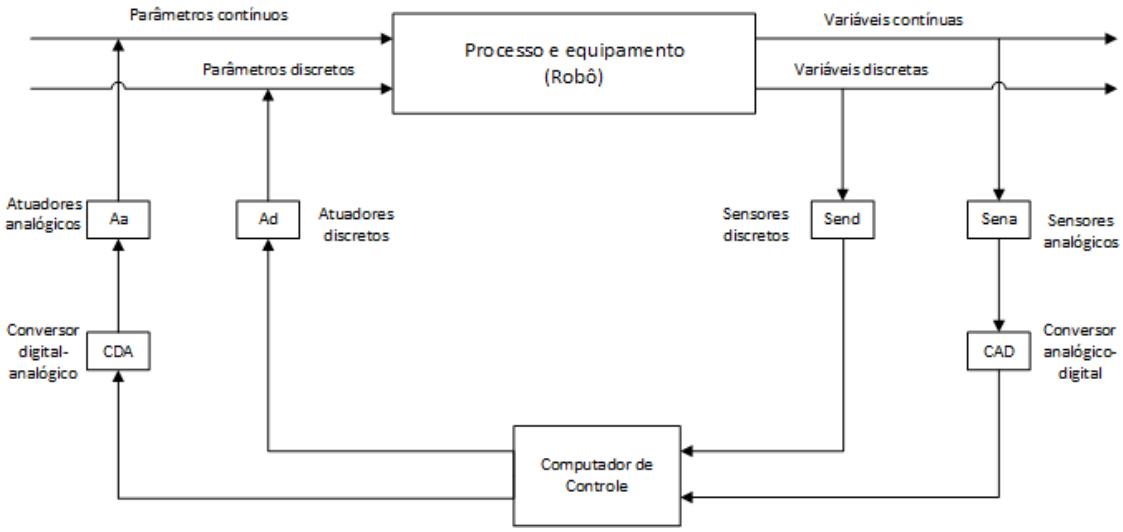


Figura 2.7: Componentes utilizados para implementação da interface.

Conforme a Figura 2.7 acima, devem ser observados que os itens 3,4 e 5 se encontram no próprio microcontrolador.

Uma variável contínua (ou parâmetro) é a que se mantém ininterrupta em um tempo de operação (execução de alguma tarefa do robô). Já uma variável discreta, pode assumir apenas certos valores em um dado intervalo, como os valores binários (zero ou um).

2.5.1 Sensores

Os Sensores são dispositivos essenciais em um sistema de controle. Eles serão utilizados no robô para dar sinais de referência sobre cada tarefa a ser executada. Segundo (Groover, 2011), um sensor é um transdutor, um dispositivo que converte uma variável física de uma forma em outra mais útil para a aplicação em questão. Isto significa que um sensor é um dispositivo que converte um estímulo físico ou uma variável de interesse (tal como uma distância de um objeto) em uma forma mais conveniente (em geral em um sinal elétrico), cujo propósito é medir o estímulo.

A finalidade dos sensores é obter variáveis de forma que consiga trata-las como valores numéricos. Os sensores podem ser classificados de diferentes formas, abaixo segue a Tabela 2.1 sobre sensores que serão usados.

Tabela 2.1: Tipos de Sensores

Categoria do estímulo	Exemplos de variáveis física
Mecânico	Posição (deslocamento, linear e angular), velocidade, aceleração, força, torque, pressão, desgaste, tensão, massa, densidade.
Elétrico	Tensão, corrente, carga, resistência, condutividade, capacitância.

Além do tipo de estímulo, os sensores também são classificados como analógicos ou discretos. Um dispositivo de medição analógico produz um sinal analógico contínuo como uma tensão elétrica, cujo o valor varia de modo analógico com a variável sendo medida. O sinal de saída de um dispositivo de medição analógico deve ser convertido em dados digitais por um conversor analógico-digital, o que no caso do robô é feito pelo o próprio microcontrolador.

Já um dispositivo de medição discreto produz uma saída que pode ter somente determinados valores. Estes dispositivos são divididos em duas categorias: binários e digitais. Um dispositivo de medição binária produz um sinal ligado/desligado. Um dispositivo de medição digital produz um sinal de saída digital tanto na forma de um conjunto de bits paralelos, como uma série de pulsos que podem ser contados, no caso deste trabalho o encoder que é um codificador ótico.

Outro fator sobre os sensores é que eles podem ser ativos ou passivos. O sensor ativo responde ao estímulo sem a necessidade de energia externa. Por outro lado, no sensor passivo é necessária uma fonte externa de energia para ele operar.

Para levantar a planta de controle do robô, deve-se encontrar a função de transferência de cada sensor utilizado, o que se faz pela relação entre o valor do estímulo físico e o valor do sinal produzido pelo sensor em resposta ao estímulo, já que a função de transferência é a relação entre o sinal de entrada pelo sinal de saída. Sendo assim, o estímulo é a entrada e o sinal gerado pelo dispositivo é a saída.

Sensores binários têm relações funcionais binários definidas pelas seguintes expressões, $S = 1$ se $S > 0$ e $S = 0$, se $S \leq 0$.

Sensores Analógicos de medição têm relações funcionais ideais em uma relação proporcional simples, pela seguinte expressão, $S = C + ms$.

Onde C é o valor de saída quando o valor do estímulo é igual a zero, e m é a constante de proporcionalidade entre s e S. A constante m pode ser definida como a sensibilidade do sensor. É uma medida de quando a saída do sensor é afetada pelo estímulo.

2.5.2 Atuadores

Nos sistemas de controle, um atuador é um dispositivo de hardware que converte um sinal de comando do controlador em uma mudança em um parâmetro físico. Segundo (Groover, 2011), um atuador é um transdutor, visto que transforma um tipo de quantidade física, como uma corrente elétrica, em outro tipo de quantidade física, como uma velocidade de rotação de um motor elétrico. Neste trabalho o atuador é o motor CC, conforme se encontra ilustrado na Figura 2.8: Desenho 3D do motor com redutor.

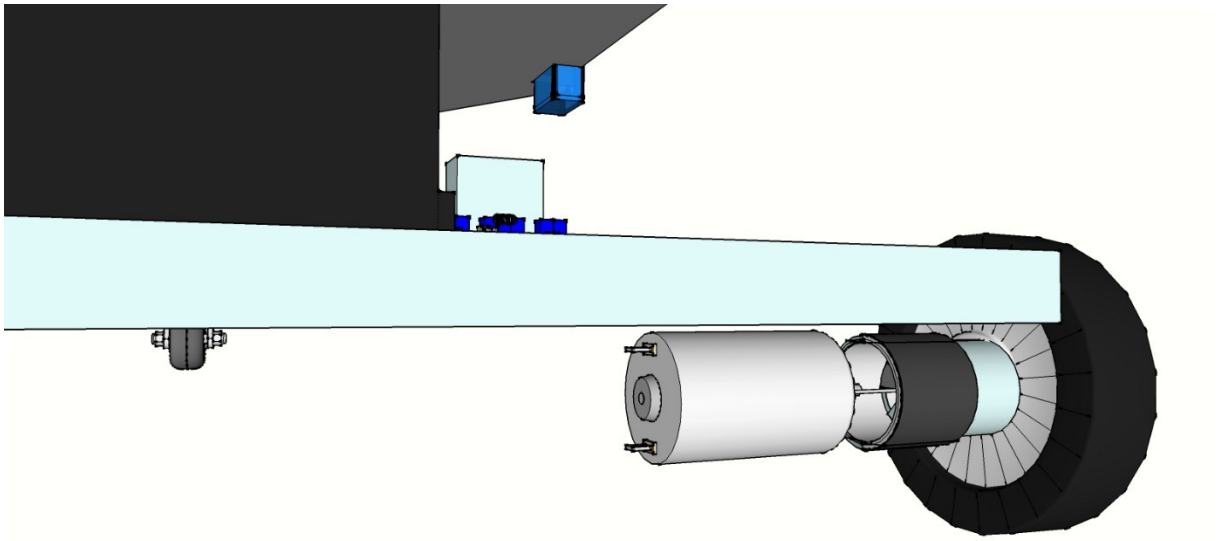


Figura 2.8: Desenho 3D do motor com redutor.

A Figura 2.8: Desenho 3D do motor com redutor. No último bloco do eixo em que se encontra a roda do robô é ilustrado o motor CC, e no primeiro bloco o redutor que está acoplado ao motor CC.

Cada roda do robô é controlada por um motor CC. Os motores CC são largamente utilizados pelo fato de ser conveniente a utilização de corrente contínua como fonte de energia. Ademais, sua relação torque/velocidade é um atrativo em muitas aplicações.

Estes motores são excitados por tensão e corrente constante. No caso do robô, a ponte H envia de acordo com o controle do microcontrolador corrente suficiente para produzir o torque que aciona o rotor. A magnitude do torque do rotor é de acordo com a corrente que circula através do enrolamento. A relação pode ser modelada através da seguinte equação:

$$T = K_t * I_a \; onde \begin{cases} T = Torque \\ I_a = Somatória\ da\ corrente\ de\ armadura \\ K_t = Torque\ constante\ do\ motor \end{cases} \quad (2.1)$$

A rotação da armadura no campo magnético do estator produz uma tensão nos terminais da armadura denominada força contraeletromotriz. Esta força aumenta com a velocidade de rotação da seguinte maneira:

$$E_b = K_v * \omega \text{ onde} \begin{cases} E_b = \text{Força contraeletromotriz} \\ K_v = \text{Tensão constante do motor} \\ \omega = \text{Velocidade Angular} \end{cases} \quad (2.2)$$

A força contraeletromotriz reduz a corrente que flui no enrolamento da armadura. Para ser calculado o rpm deste motor em função da velocidade (ω), se utiliza a seguinte equação:

$$RPM = \frac{60*\omega}{2\pi} \quad (2.3)$$

Tendo em mãos os valores das características do motor CC é possível calcular a corrente da armadura pela seguinte equação:

$$Ia = \frac{V_{in}-E_b}{R_a} \begin{cases} V_{in} = \text{Tensão de entrada(terminais do motor DC)} \\ R_a = \text{Resistência da armadura} \end{cases} \quad (2.4)$$

A corrente real na armadura, que também é a corrente que produz o torque inicial do motor, depende da velocidade de rotação do rotor, uma vez que a força contraeletromotriz tende a diminuir a corrente de armadura e esta força é maior quanto maior for a velocidade angular da rotação do rotor.

No caso do Robô, o motor utilizado é do fabricante CUI INC modelo M233. Suas especificações se encontram no Anexo C deste trabalho. A Figura 2.9: Motor M233 com redutor e encoderilustra o motor M233 utilizado na montagem.

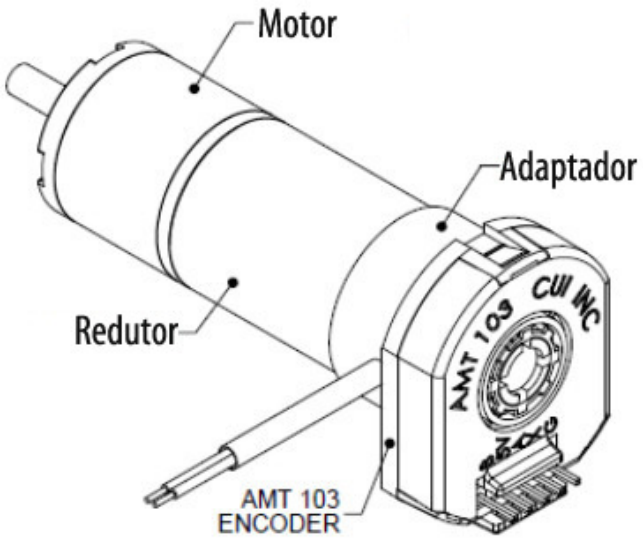


Figura 2.9: Motor M233 com redutor e encoder.

Condutores vistos na Figura 2.9, são conectados na saída dos terminais do L298 que fornecerá tensão suficiente para operação dos motores no robô.

2.5.3 Contadores de pulsos (Encoder)

Encoder converte o movimento circular de um eixo mecânico em pulsos elétricos de modo que estes pulsos possam ser utilizados como sinais de entrada em um microcontrolador para medição de velocidade do eixo ao qual o encoder está acoplado.

Um encoder é um dispositivo para medição de velocidade de rotação, que consiste de uma fonte de luz e fotodetectores em cada lado de um disco. O disco contém ranhuras uniformemente distribuídas ao longo da parte exterior de sua face. Essas ranhuras permitem a passagem da luz e a energização do fotodetector. O disco é conectado ao eixo de rotação cuja posição angular e velocidade devem ser determinados. Conforme o eixo gira, as ranhuras fazem com que a luz seja detectada pelo fotodetector com uma série de flashes. Os flashes são convertidos em pulsos elétricos e, pela contagem dos pulsos e o respectivo cálculo da frequência, pode-se determinar a velocidade do motor DC de cada roda do robô.

Segue abaixo a figura desenhada do encoder no robô. A Figura 2.10 ilustra a posição dos encoder no robô, que se encontra no último bloco do eixo em que está a roda do robô.

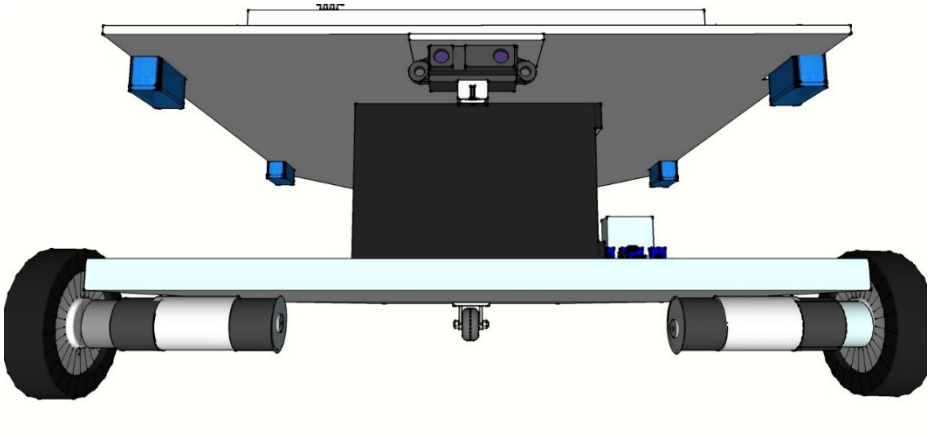


Figura 2.10: Desenho 3D encoder.

O Encoder utilizado no robô é do fabricante CUI INC modelo AMT103, cujas especificações do modelo seguem no Anexo B deste trabalho. A Figura 2.11 apresenta imagens do encoder e suas ligações.

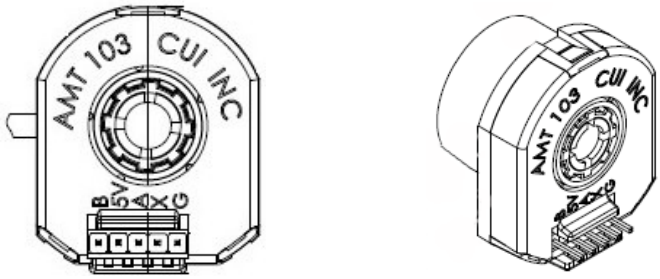


Figura 2.11: Imagem do encoder cedida pelo fabricante.

As ligações do encoder foram feitas da seguinte forma no microcontrolador:

Tabela 2.2: Ligações do Encoder

Tipo	Encoder	Microcontrolador
Ligação	B	Portas de Interrupções
Ligação	5V	5V
Ligação	A	Portas de Interrupções
Ligação	X	Ø

Ligação	G	GND

2.5.4 Ponte H

Neste trabalho, utiliza-se o circuito integrado L298 que utiliza 2 pontes completas independentes (full-bridge drive), onde o objetivo é ativar o motor CC, uma vez que no projeto são usados motores de 12 V e a saída do microcontrolador fornece um valor máximo de 5V. Foram empregadas duas pontes H, com o objetivo de controlar os sentidos de rotação dos motores e, por consequência, o sentido da corrente que circula entre os polos dos motores, visto que os motores CC alteram seus sentidos de rotação quando inverte-se sua polaridade. Na Figura 2.12 pode se observar uma imagem em 3D da ponte H no robô.

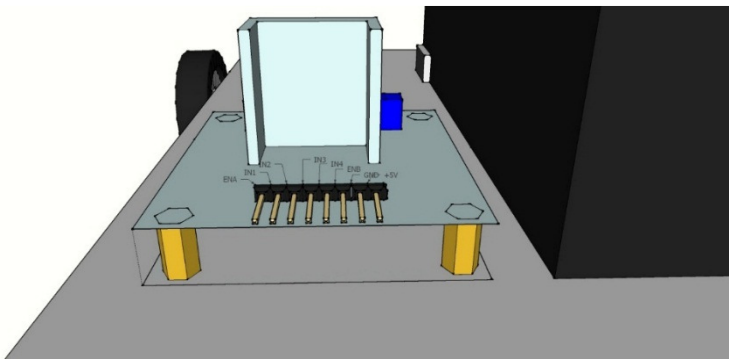


Figura 2.12: Desenho 3D do circuito integrado L298.

Na Figura 2.12, cada ponte H do L298 é controlada por 3 entradas, onde 1 entrada é de habilitação (EnA) que ativa ou desativa a ponte H e as outras 2 (In1 e In2) determinam a circulação interna da corrente. Isto também é válido para a outra ponte H. (EnB, In3 e In4). A Tabela 2.3 apresenta o funcionamento de uma ponte H.

Tabela 2.3: Tabela logica para o comando da L298

EnA	In1	In2	Motor
1	1	0	Giro sentido horário
1	0	1	Giro sentido anti-horário
0	X	X	Parado
X	1	1	Parado

X	0	0	Parado

Tabela 2.3 ilustra X como ausência de valor.

2.5.5 Microcontrolador Atmega 2560

Este é o microcontrolador que possui uma cpu de 8 bit AVR que será utilizado no robô, conforme ilustra a Figura 2.13.

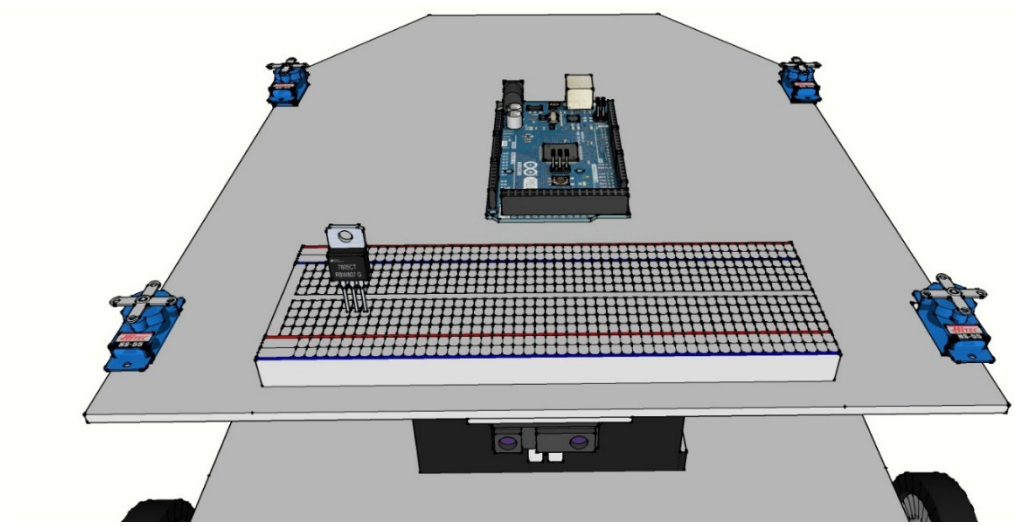


Figura 2.13: Desenho 3D Arduino mega 2560.

A escolha da placa Arduino é pelo fato de ser uma placa de desenvolvimento aberta, ter baixo custo e também por facilitar a criação de projetos eletrônicos. De acordo com o fabricante a placa Arduino Mega 2560 possui 54 pinos de entradas/saídas digitais, 16 entradas analógicas, 4 UARTs(porta seriais de hardware), um oscilador de cristal de 16MHZ, uma conexão USB, uma entrada de alimentação, uma conexão ICSP e um botão de reset, como se vê na Figura 2.14.

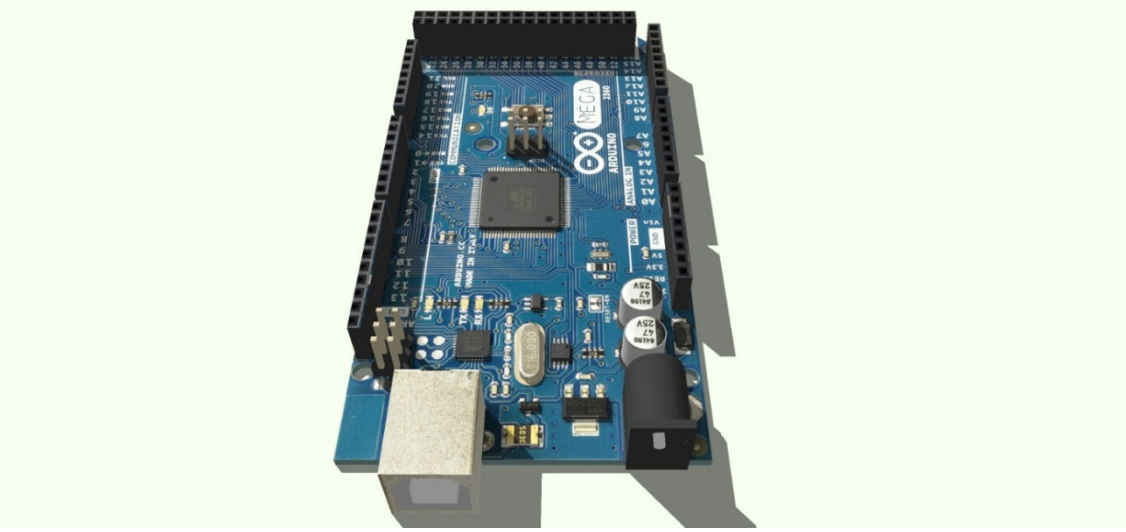


Figura 2.14: Arduino Mega 2560.

Das portas de entrada e saída citadas acima, este microcontrolador possui 6 pinos de interrupções externas, onde 1 par será usado para entrada do sensor do encoder de cada motor restando ainda 4 pinos de reserva. A conexão ICSP será usada para conexão de um micro SD, para gravar os valores gerados pelos encoder quando o robô estiver em movimento e podê-lo comparar aos resultados simulados. Para o controle da velocidade do motor CC serão empregadas duas saídas PWM (modulação por largura de pulso), uma para cada motor.

O PWM é uma técnica utilizada por sistemas digitais, para variação do valor médio de uma forma de onda periódica. Esta técnica tem por finalidade manter a frequência de uma onda quadrada fixa e ao mesmo tempo variar o interregno que o sinal fica em nível lógico alto, sendo este período conhecido como duty cycle, que é o ciclo ativo da forma de onda, conforme pode ser verificado no Anexo D.

No Anexo D, nota-se que a frequência da forma de onda para 0% duty ciclo tem o mesmo valor para 25%,50%,75% e 100% do duty cyclo, variando somente o duty cycle da forma de onda. Esta frequência é cerca de 500 Hz para o PWM do Arduino. De acordo com (Lima e Villaça, 2012), o cálculo do valor médio de um sinal digital é dado pela equação 2.5

$$V_{Médio} = \frac{Amplitude\ Máxima}{Período} * (Tempo\ Ativo\ no\ Período) \quad (2.5)$$

Desta forma quando o duty cicle está em 100% do valor médio da saída assume o seu valor máximo, ou seja 5 V, e quando o duty cicle está em 0% assume seu valor mínimo, ou seja, 0 V. O mesmo ocorre para um ciclo ativo de 50% que corresponde a um valor médio de

2,5 V e também um ciclo ativo de 75% correspondente a 3,75V. Assim, alterando o ciclo útil do sinal PWM altera-se o seu valor médio.

A Resolução do PWM em um microcontrolador é dada pelo seu número de bits e indica quantos diferentes ciclos ativos podem ser gerados. Como o microcontrolador utilizados tem um cpu de 8 bits, ele pode gerar 256 diferentes níveis, começando com o nível de tensão de 0 e terminando com 100% do ciclo ativo. A contagem se dá de 0 até 255 num ciclo contínuo.

Na Figura 2.15 é apresentada a foto tirada do robô móvel, ilustrando-se seus componentes.

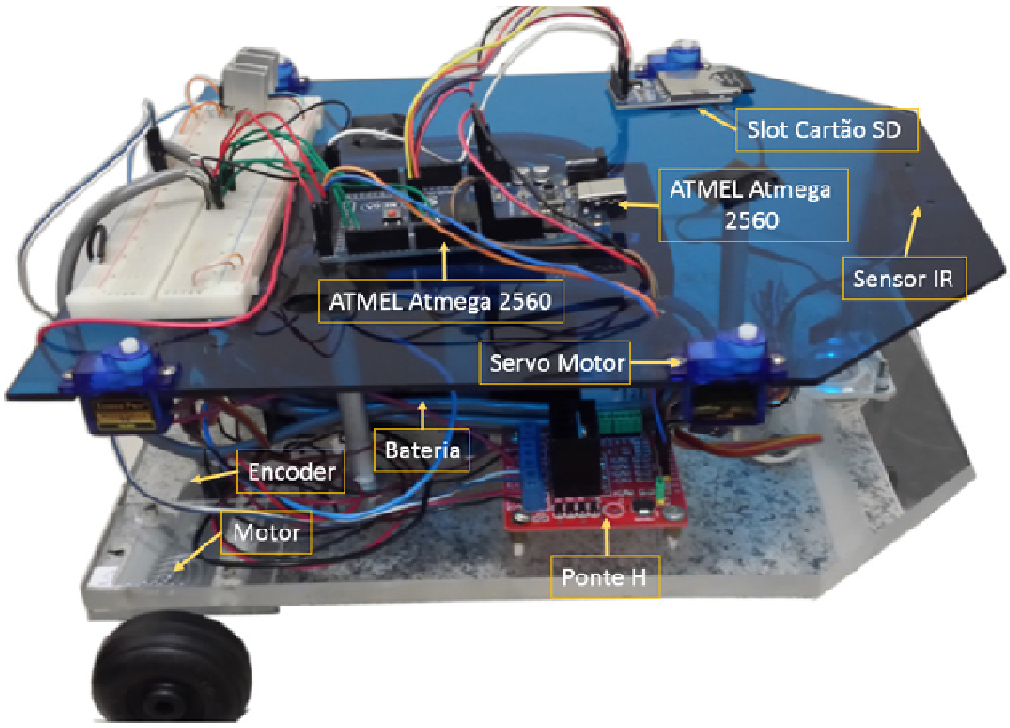


Figura 2.15: Foto do robô móvel

2.6 Considerações do Capítulo

Foram apresentados os projetos em malha aberta e fechada para o controle do sistema robótico deste trabalho, além da apresentação dos componentes de hardware que serão utilizados para este projeto.

No capítulo seguinte será desenvolvido o modelo cinemático e dinâmico para controle do sistema e implementação das trajetórias do robô.

3 MODELAGEM CINEMÁTICA E DINÂMICA

Neste Capítulo será apresentado o desenvolvimento da cinemática direta e dinâmica do robô.

A Modelagem cinemática e dinâmica do robô opera-se com base no estudo desenvolvido por (MELO, 2007), isto significa que, em princípio, o s dados recolhidos das características do robô serão aplicados na teoria e simulados no MatLab para verificar a autenticidade dos resultados.

Posteriormente, será realizado o modelo dinâmico e verificados os resultados aplicando a teoria diretamente no algoritmo do controlador do robô e, com base nos dados obtidos da leitura do encoder sobre a trajetória percorrida pelo robô, será analisado se é igual ao valor teórico ou próximo dele para comprovação da correta aplicação da teoria. A Figura 3.1 apresenta o fluxograma do desenvolvimento da cinemática para o robô.

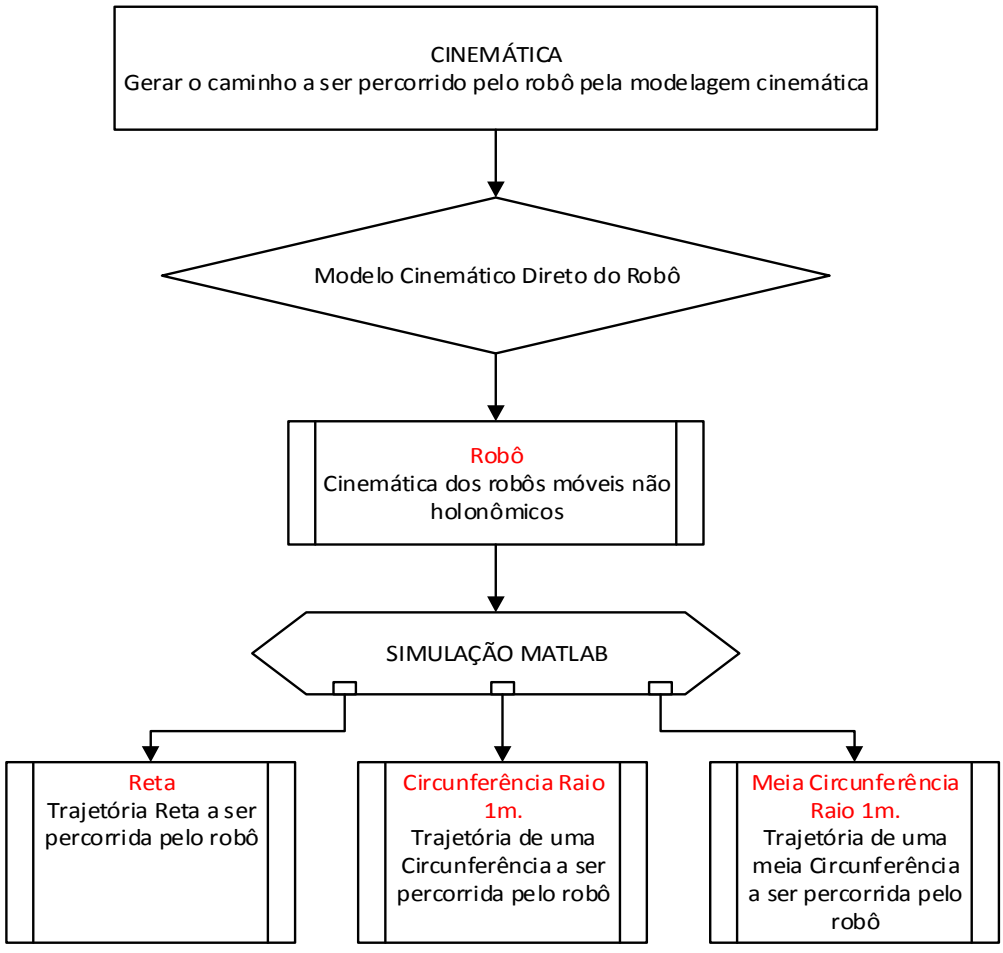


Figura 3.1: Fluxograma do desenvolvimento da cinemática para o robô.

3.1 Modelo Cinemático Direto Do Robô

Segundo (Melo, 2007) a determinação dos posicionamentos possíveis de serem alcançados, dados os parâmetros de controle, é conhecido como problema da cinemática direta para o robô.

De acordo com o modelo do robô disponibilizado para o estudo, já citado no trabalho de (BORGES, 2014), é necessário encontrar e analisar sua posição no espaço. Conforme visto no capítulo de controle, se consegue fazer o controle da direção do robô através da velocidade aplicada em cada roda, porém somente a malha de controle não é suficiente para fazer o controle da direção, é necessário também um modelo cinemático. Desta forma, o robô é inserido em um plano qualquer onde é referenciado através dos eixos X e Y conforme pode ser visto na Figura 3.2.

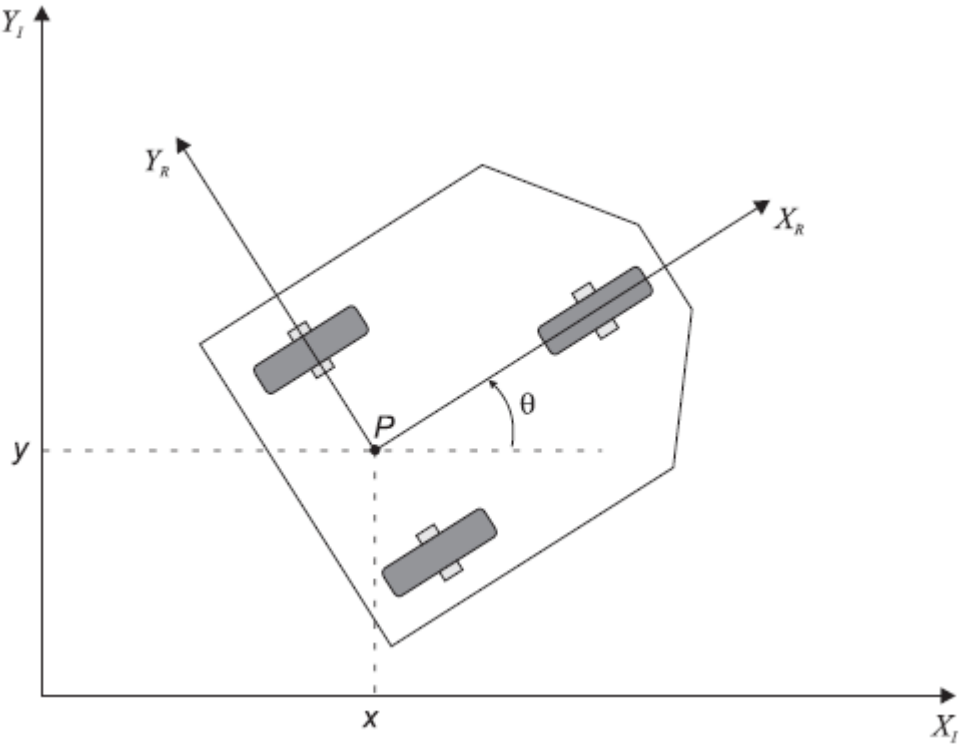


Figura 3.2: Plano de referência global e a referência local do robô.

Assim os eixos X_I e Y_I definem uma base inercial arbitrária no plano cartesiano. Para se especificar a posição do robô, escolher-se um ponto P no chassi do robô, como visto na Figura 3.2. Tendo o ponto P como referência são traçados mais dois eixos no local sendo o ponto P a origem deste plano, o qual define-se como ponto de referência local. Correlacionando o plano de referência local com o plano cartesiano, os planos de referência

global construídos através dos eixos x e y conforme a Figura 3.2, tendo como referência o ponto P a origem e seus eixos correspondentes como destino. Assim, procede a seguinte situação de mapeamento do robô.

Tabela 3.1: Planos de Referência do robô.

Eixo X	Eixo Y	Plano
X_I	Y_I	Cartesiano
x	y	Global
X_R	Y_R	Local

Tendo os planos como base, o ângulo θ é encontrado a partir da diferença angular entre o plano de referência local e global. Desta forma a posição do robô pode ser dado pelo vetor ξ, onde se referencia a posição do robô no eixo global, ficando:

$$\xi_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{3.1}$$

A partir desta equação, é necessário mapear a movimentação do robô ao longo dos eixos do plano de referência global para o plano de referência local. Para isso, essencial utilizar a matriz de rotação ortogonal que é definida por:

$$R(\theta) = \begin{bmatrix} cos\ (\theta) & sen(\theta) & 0 \\ -sen(\theta) & cos\ (\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

A matriz R(θ) desempenha o papel de mapear a movimentação do robô no plano de referência global para o plano de referência local. Esta operação também depende do valor de "θ",então a relação que descreve este mapeamento é dado por (SIEGWART e NOURBAKHSH, 2004)

$$\xi_R = R(\theta) * \xi_I \tag{3.3}$$

Desta forma, esta equação descreve a cinemática direta do robô móvel em um plano cartesiano. Este modelo representa um robô móvel com tração diferencial por meio de duas rodas traseiras e uma roda livre(castor) possuindo um diâmetro "d", com um espaçamento L, conforme a Figura 3.3.

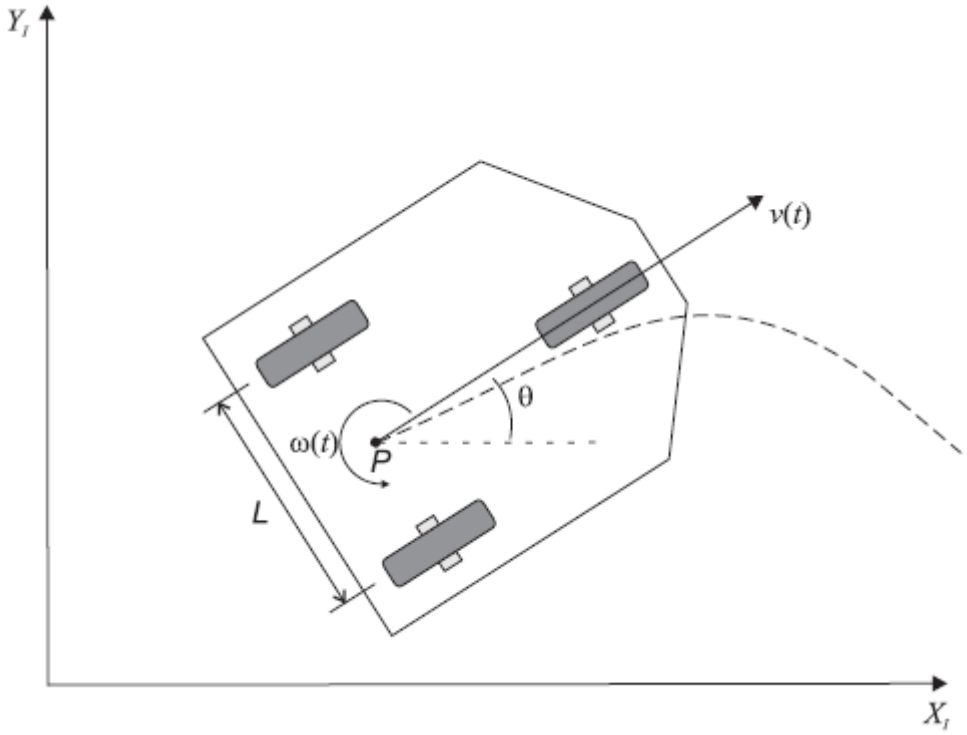


Figura 3.3: Robô móvel com tração diferencial no plano global de referência.

Segundo (MELO, 2007), um modelo de cinemática direta que consegue descrever a velocidade do robô no plano geral de referência, o que pode ser dado por:

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \begin{cases} Variável\ instantanea\ de\ x \\ Variável\ instantanea\ de\ y \\ Variável\ instantanea\ de\ \theta \end{cases} \tag{3.4}$$

Desta forma, é possível calcular a variação do movimento do robô no plano de referência global com base na variação do movimento em seu plano local. Assim, utilizando a mesma dedução que já foi citada, encontra-se a posição do robô no plano global.

$$\dot{\xi}_R = R(\theta) * \dot{\xi}_I \tag{3.5}$$

$$\dot{\xi}_I = \dot{\xi}_R / R(\theta) \tag{3.6}$$

$$\dot{\xi}_I = R(\theta)^{-1} * \dot{\xi}_R \tag{3.7}$$

Para o plano de referência Local, se o robô estiver se movendo em linha reta em direção a componente $+X_R$ já mostrada aqui, as velocidades das rodas direita e esquerda podem ser equacionadas da seguinte forma:

$$V_d = \dot{x}_{R1} = d\frac{\dot{\varphi}_1}{2} \tag{3.8}$$

$$V_e = \dot{x}_{R2} = d\frac{\dot{\varphi}_2}{2} \tag{3.9}$$

Somando o movimento de cada roda, é possível determinar as variáveis $\dot{x}_R$ e ξ_R. Considerando que não haja escorregamentos laterais, $\dot{y}_R$ é sempre igual a zero.

O raio de curvatura que o robô desenvolve no momento em que as velocidade das duas rodas são distintas, é igual a:

$$R_C = \frac{L(V_D + V_E)}{2*(V_D - V_E)} \tag{3.10}$$

3.2 Sistema de Transmissão Direta

Como já citado neste trabalho, a transmissão deste robô consiste de duas rodas montadas sobre o mesmo eixo de tração sendo controladas por motores individuais. A cinemática trata da relação entre os parâmetros de controle e o comportamento do sistema no espaço.

Considerando que o controle da velocidade das rodas determina o movimento do robô, no caso de transmissão diferencial, para que o robô execute uma curva, este deve rodar em torno de um ponto que se localiza no eixo comum das duas rodas. Variando a velocidade relativa das duas rodas, o ponto cujo qual o robô executa a rotação pode variar, escolhendo-se diferentes trajetórias.

A cada instante de tempo, o ponto de rotação do robô tem que ter a propriedade que faz com que a roda esquerda e a direita sigam um caminho ao redor do CCI com a mesma taxa de velocidade angular "ω", conforme ilustrado na Figura 3.4.

$$V_d = \omega(R+L/2) \tag{3.11}$$

$$V_e = \omega(R-L/2) \tag{3.12}$$

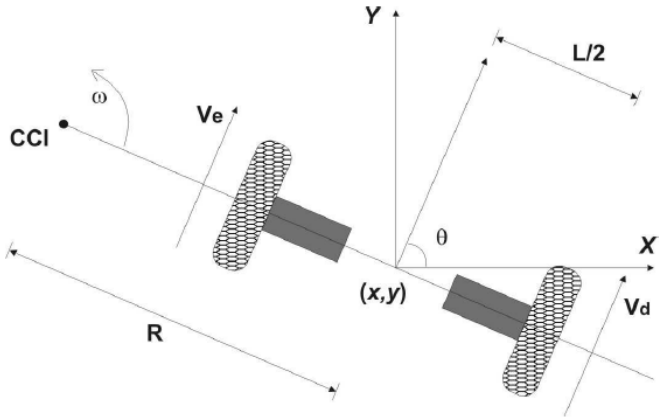


Figura 3.4: A cinemática de transmissão diferencial num robô móvel.

Assim denominam-se abaixo as seguintes constantes e variáveis do sistema, vistos na Tabela 3.2.

Tabela 3.2: Variáveis e Constantes do robô

Constantes ou Variável	Representação
L	Espaçamento entre as rodas do centro de cada roda
R	Distância do CCI até o ponto central entre as rodas
Θ	Ângulo de orientação
v_e	Velocidade linear da roda esquerda
v_d	Velocidade linear da roda direita

Desta forma, se forem resolvidas estas equações para R e posteriormente para ω, tem-se:

$$\frac{v_d}{R+\frac{L}{2}} = \frac{v_e}{R-\frac{L}{2}} \tag{3.13}$$

Rearranjando

$$R = \frac{L}{2}\,\frac{(v_d+v_e)}{2\,(v_d-v_e)} \tag{3.14}$$

Já para ω

$$\frac{v_d}{\omega} - \frac{L}{2} = \frac{v_e}{\omega} + \frac{L}{2} \tag{3.15}$$

Rearranjando

$$\omega = \frac{v_d - v_e}{L} \tag{3.16}$$

Algumas observações relevantes feitas por (MELO, 2007) são:

Se $V_e = V_d$, o raio R é infinito e o robô se move em linha reta;

Se $V_e = -V_d$, então o raio é zero e o robô roda sob um ponto central entre as duas rodas(ele não se desloca linearmente);

Para outros valores de V_e e V_d o robô não se moverá em linha reta e sim seguirá uma trajetória curva ao redor de um ponto de distância R do centro do robô.

Obs.: Pequenos erros da velocidade enviados a cada roda do motor resultam em diferentes trajetórias. Veículos com transmissão diferencial são muito sensíveis a velocidade relativa da duas rodas.

3.3 Cinemática Direta para Transmissão Diferencial

Neste ponto será analisada a hipótese em que se conhece a posição onde o robô se encontra.

Para um robô que se encontra em um determinado ponto (x,y) e está direcionado com um ângulo θ com o eixo x conforme ilustra a Figura 3.5.

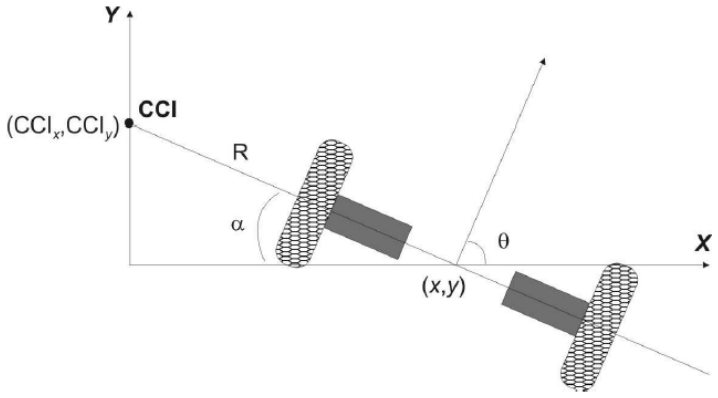


Figura 3.5: Geometria de cinemática direta num robô com transmissão diferencial.

Manipulando V_e e V_d pode-se ter diferentes posicionamentos. Aqui o robô se encontra na posição (x,y, θ) resultando em:

$$\alpha = \frac{\pi}{2} - \theta \tag{3.17}$$

Pelas relações trigonométricas do sistema, fica:

$$CCI_x = x - R\ cos\ (\alpha) \tag{3.18}$$

$$CCI_y = y - R\ sen\ (\alpha) \tag{3.19}$$

Onde substituindo alfa

$$CCI_x = x - R\ cos\ (\frac{\pi}{2} - \theta) \tag{3.20}$$

$$CCI_y = y - R\ sen\ (\frac{\pi}{2} - \theta) \tag{3.21}$$

Resultando

$$CCI_x = x - R\ sen(\theta) \tag{3.22}$$

$$CCI_y = y + R\ cos(\theta) \tag{3.23}$$

Desta forma, expandindo-se a função para ser colocada em função do tempo, assim tem-se que:

Para instante de tempo t onde o robô é presentado por P(x,y, θ) tem-se:

$$R = \frac{x - CCI_x}{sen\ (\theta)} \tag{3.24}$$

$$R = -\frac{y - CCI_y}{cos(\theta)} \tag{3.25}$$

Para um instante de tempo t = t + δt, tem-se P(t+ δt) = (x', y', θ') e como R e CCI são constantes, fica

$$CCI_x = x' - R\ sen\ (\theta') \tag{3.26}$$

$$CCI_y = y' + R\ cos\ (\theta') \tag{3.27}$$

A Figura 3.6 ilustra o desenvolvimento das equações acima.

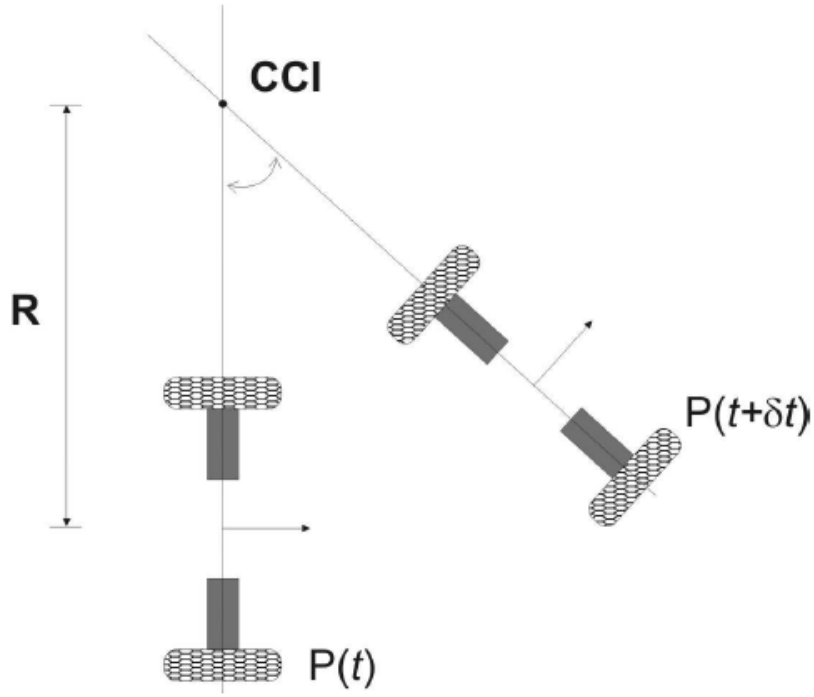


Figura 3.6: Cinemática direta num robô com transmissão diferencial em t = t + δt.

Sabendo-se que o deslocamento angular no instante $t = t + \delta t$ é igual a $\omega\delta t$, tem-se:

$$\theta' = \theta + \omega\delta t \tag{3.28}$$

Logo, tendo o ângulo θ' e o raio *R*, substituo tudo na equação anterior, resultando em:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + v\ cos\ (\theta)\delta t \\ y + v\ sen(\theta)\delta t \\ \theta \end{bmatrix} \tag{3.29}$$

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} cos\ (\omega\delta t) & -sen(\omega\delta t) & 0 \\ sen(\omega\delta t) & cos\ (\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - CCI_x \\ y - CCI_y \\ \theta \end{bmatrix} + \begin{bmatrix} CCI_x \\ CCI_y \\ \omega\delta t \end{bmatrix} \tag{3.30}$$

Desta maneira, se o robô está na posição (x,y, θ) no tempo t então no instante t = t + δt a posição do robô será encontrada pela equação acima.

A equação acima descreve o movimento rotacional do robô a uma distância R do CCI com uma velocidade angular "ω" (DUDEK e JENKIN, 2000).

3.3.1 Encontrando o posicionamento do robô para qualquer tempo "t" baseando-se nas velocidades de suas rodas.

É possível resolver o problema da cinemática direta para o sistema integrando a equação anterior a partir das condições iniciais ($x_0,\ y_0,\ \theta_0$), caso não ocorra patinação.

$$x(t) = \int_0^t V(t)\, cos[\theta(t)]dt \tag{3.31}$$

$$y(t) = \int_0^t V(t)\, sen[\theta(t)]dt \tag{3.32}$$

$$\theta(t) = \int_0^t \omega(t)dt \tag{3.33}$$

Para este caso com transmissão diferencial, tem-se:

$$x(t) = \frac{1}{2}\int_0^t [v_d(t) + v_e(t)]\, cos[\theta(t)]\, dt \tag{3.34}$$

$$y(t) = \frac{1}{2}\int_0^t [v_d(t) + v_e(t)]\, sen[\theta(t)]\, dt \tag{3.35}$$

$$\theta = \frac{1}{L}\int_0^t [v_d(t) - v_e(t)]dt \tag{3.36}$$

3.3.2 Cinemática dos robôs móveis não holonômicos

A segunda hipótese, estudada neste tópico, ocorre quando a velocidade do robô não pode ser integrada num determinado posicionamento.

$$V_d \neq V_e \tag{3.37}$$

Para este caso onde não é possível integrar a velocidade do robô é admitida as seguintes situações:

$$V_e(t) = V_e \tag{3.38}$$

$$V_{d(t)} = V_d \tag{3.39}$$

$$x(t) = \frac{L}{2}\frac{v_d + v_e}{v_d - v_e}\, sen\left[\frac{t}{L}\,(v_d - v_e)\right] \tag{3.40}$$

$$y(t) = -\frac{L}{2}\frac{v_d + v_e}{v_d - v_e}\, cos\left[\frac{t}{L}\,(v_d - v_e)\right] + \frac{L}{2}\frac{v_d + v_e}{v_d - v_e} \tag{3.41}$$

$$\theta(t) = \frac{t}{L}\,(v_d - v_e) \tag{3.42}$$

Onde $(x, y, \theta)_{t=0}$ = (0,0,0). Esta equação não fornece um controle independente em relação a θ, desta forma o robô estará sempre se deslocando sobre o mesmo raio R que passa através do ponto (0, 0) no instante t=0.

$$V_d = V_e \tag{3.43}$$

Neste caso o robô se movimenta em linha reta com raio R tendendo ao infinito, desta forma é possível usar a seguinte equação.

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + v\ cos\ (\theta)\delta t \\ y + v\ sen(\theta)\delta t \\ \theta \end{bmatrix} \tag{3.44}$$

Caso -V_e = V_d, o robô irá rodar sobre seu eixo, sem deslocamento linear, resultando na seguinte equação (DUDEK e JENKIN, 2000).

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta + \frac{2v\delta t}{L} \end{bmatrix} \tag{3.45}$$

Sendo esta uma técnica para posicionar o robô em algum ponto específico (x,y, θ) até que ele esteja posicionado no sentido do ponto (x,y) e atinja a orientação θ desejada.

3.4 Modelagem Dinâmica do Robô

O diagrama do desenvolvimento da Dinâmica do Robô é mostrado na Figura 3.7.

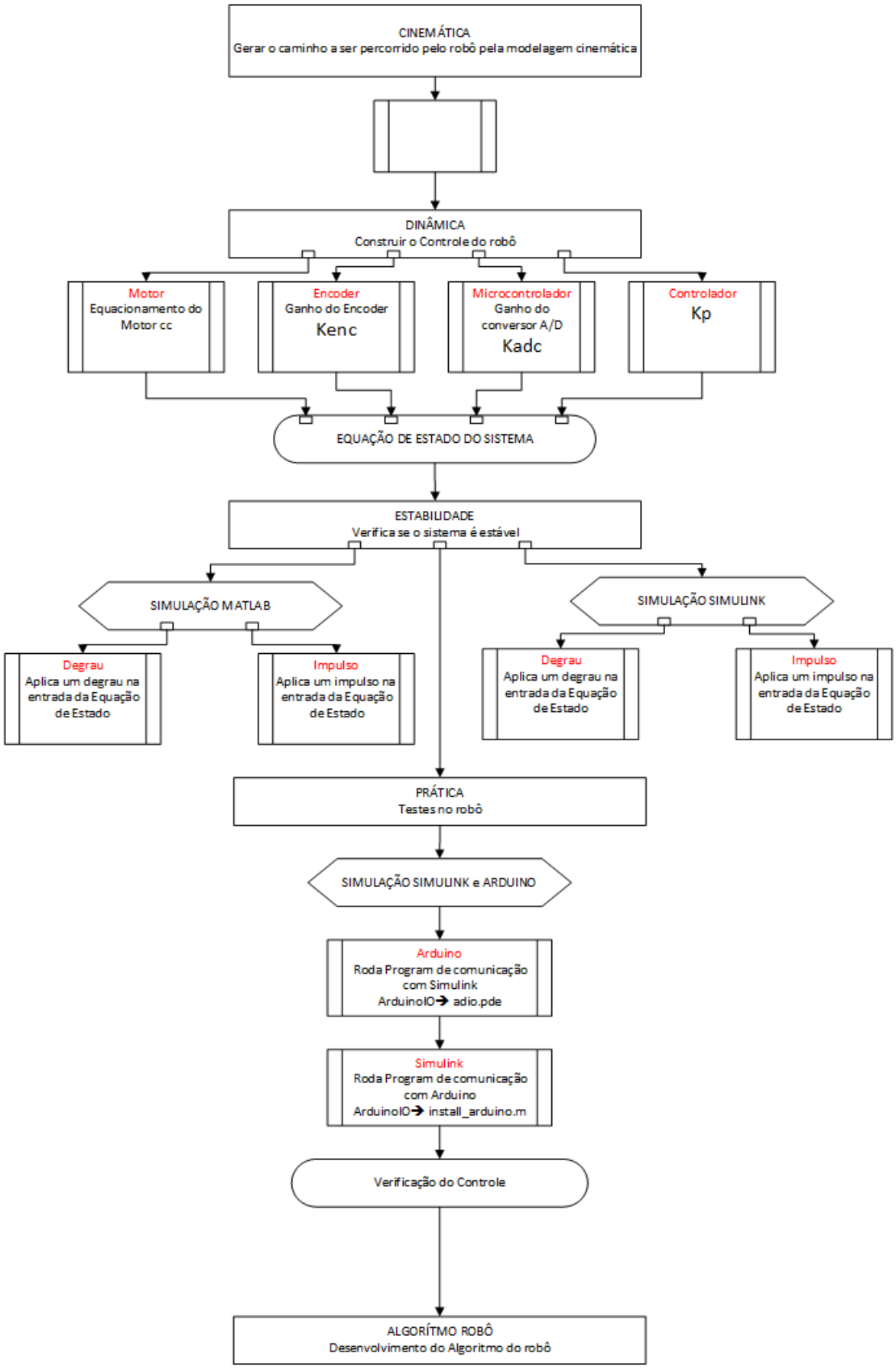


Figura 3.7: Fluxograma do modelo dinâmico desenvolvido para o robô.

Na dinâmica e controle o objetivo é comandar adequadamente os motores para cada roda em situações distintas de atrito da roda com o solo. Portanto, requer-se um controle que consiga mantem fixa a velocidade de rotação dos motores independentemente da região que é percorrida.

Para a solução deste caso, é necessário utilizar um controlador. Assim precisamos calcular a função de transferência do controlador e do motor. Neste trabalho iremos equacionar o sistema pelo método espaço de estado.

3.4.1 Sistema no espaço de estado

$$\dot{x} = \mathbf{A}\mathrm{x}(\mathrm{t}) + \mathbf{B}\mathrm{u}(\mathrm{t}) \tag{3.46}$$

$$y = \mathbf{C}\mathrm{x}(\mathrm{t}) + \mathbf{D}\mathrm{u}(\mathrm{t}) \tag{3.47}$$

A = Matriz do sistema descreve como o estado atual afeta a alteração do estado $\dot{x}$;

B = Matriz de controle indica como a entrada de controle afetará a mudança de estado;

C = Determina a relação do sistema e a saída

D = Matriz da entrada do controle

Em uma implementação do sistema de controle em um microcontrolador é vantajoso usar a formulação de espaço de estado, com o modelo representado na forma discreta.

Usando o Espaço de estado é fácil incluir no sistema os:

Ganho do Conversor Analógico-Digital

$$K_{ADC} = \frac{5\mathrm{V}}{2^{10}} = 4{,}88\mathrm{mV} \begin{cases} \text{Atmega 2560} \\ \text{Conversor AD} = 10 \text{ bits} \end{cases} \tag{3.48}$$

Obs.: Usado para quando quiser fazer a leitura de algum sensor no pino de entrada de sinal analógico do microcontrolador, como a leitura do encoder, ou saída, para quando quiser acionar ou controlar uma carga por exemplo como o motor cc.

Ganho do Encoder

$$K_{ENC} = \frac{96 \text{ leitura}}{2\pi} = 15{,}28 \text{ leitura/radiano} \tag{3.49}$$

3.4.2 Equacionamento do motor CC

Para modelar a equação de estado, é necessário conhecer alguns parâmetros fundamentais do motor DC que são:

Características Mecânicas do Motor CC

Características Elétricas do Motor CC

A Figura 3.8 (Control Tutorials for MatLab & Simulink) mostra o modelo elétrico do motor CC.

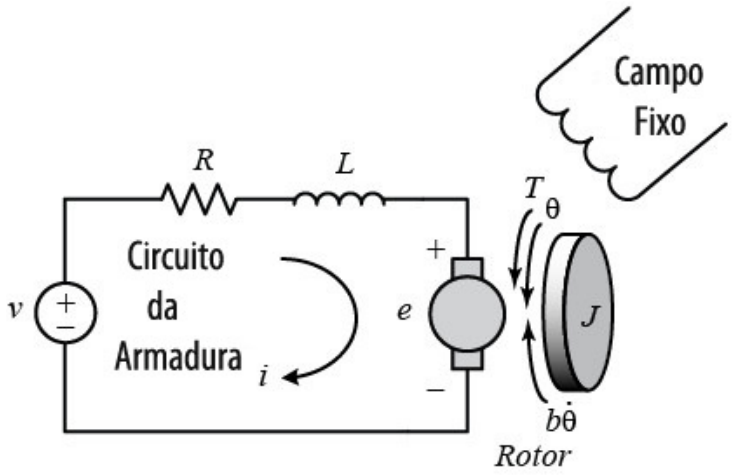


Figura 3.8: Motor CC.

Características elétricas do motor CC

Aplicando As Leis de Kirchoff

$$Vm(t) = i(t)*Ra + La*\frac{di(t)}{dt} + Ve(t) \tag{3.50}$$

Onde

$$Vm(t) = K_{pwm}*V_{pwm}(t) \begin{cases} K_{pwm} = K_{ADC} \\ V_{pwm} = Ka * V_{bateria} \end{cases} \tag{3.51}$$

$$Ve(t) = Kb*\omega(t) \tag{3.52}$$

Tendo equacionado a parte elétrica, adotaremos os seguintes passos para modelagem do sistema:

- Definição das variáveis de Estados.

- Derivação das variáveis de Estado.

De acordo com (DORF; BISHOP, 2010), as variáveis de Estado serão:

$$i(t) \tag{3.53}$$

$$\theta(t) \tag{3.54}$$

$$w(t) \tag{3.55}$$

Desta forma para a primeira situação tem-se

$$x1(t) = i(t) \tag{3.56}$$

$$x2(t) = \theta(t) \tag{3.57}$$

$$x3(t) = w(t) \tag{3.58}$$

Uma vez que sabemos qual a variável de estados, iremos deriva-las seguindo a ordem:

$$\frac{di(t)}{dt} \tag{3.59}$$

Característica elétrica do motor

$$Vm(t) = i(t) * Ra + La * \frac{di(t)}{dt} + Ve(t) \tag{3.60}$$

$$K_{pwm} * V_{pwm}(t) = i(t) * Ra + La * \frac{di(t)}{dt} + Kb * \omega(t) \tag{3.61}$$

$$La * \frac{di(t)}{dt} = -i(t) * Ra - Kb * \omega(t) + K_{pwm} * V_{pwm}(t) \tag{3.62}$$

$$\frac{di(t)}{dt} = \frac{-i(t) * Ra - Kb * \omega(t) + K_{pwm} * V_{pwm}(t)}{La} \tag{3.63}$$

$$\frac{di(t)}{dt} = \frac{-i(t) * Ra - Kb * \omega(t) + K_{ADC} * V_{bateria}(t)}{La} \tag{3.64}$$

$$\frac{d\theta(t)}{dt} \tag{3.65}$$

Característica mecânica do motor

$$\frac{d\theta(t)}{dt} = \omega(t) \tag{3.66}$$

$$\frac{d\omega(t)}{dt} \tag{3.67}$$

Característica mecânica do motor

$$T(t) = Kt * ia(t) \tag{3.68}$$

$$T(t) = Jt * \alpha(t) + b * \omega(t) \tag{3.69}$$

$$T(t) = Jt * \frac{d\omega(t)}{dt} + b * \omega(t) \tag{3.70}$$

$$Jt * \frac{d\omega(t)}{dt} = T(t) - b * \omega(t) \tag{3.71}$$

$$Jt * \frac{d\omega(t)}{dt} = Kt * ia(t) - b * \omega(t) \tag{3.72}$$

$$\frac{d\omega(t)}{dt} = \frac{Kt*ia(t) - b*\omega(t)}{Jt} \tag{3.73}$$

Uma vez derivadas as variáveis de estado, é possível montar a equação de estado do sistema.

3.4.3 Equação de estado do sistema

De acordo com (NISE, 2012), um sistema é representado no espaço de estados pelas seguintes equações:

$$\dot{x} = \mathbf{A}\mathrm{x}(\mathrm{t}) + \mathbf{B}\mathrm{u}(\mathrm{t}) \tag{3.74}$$

$$\mathrm{y} = \mathbf{C}\mathrm{x}(\mathrm{t}) + \mathbf{D}\mathrm{u}(\mathrm{t}) \tag{3.75}$$

Onde x é o vetor de estado, ẋ é a derivada do vetor de estado em relação ao tempo, y é o vetor de saída, u é o vetor de entrada, A é a matriz do sistema, B é a matriz de entrada, C é a matriz de saída e D é a matriz de transmissão direta.

Desta forma vamos segue-se a seguinte ordem para achar a equação de estado:

A. Estado do sistema em relação a "$\dot{x}$"

$$\dot{x} = \begin{bmatrix} \frac{di(t)}{dt} \\ \frac{d\theta(t)}{dt} \\ \frac{d\omega(t)}{dt} \end{bmatrix} \tag{3.76}$$

$$\begin{bmatrix} \frac{di(t)}{dt} \\ \frac{d\theta(t)}{dt} \\ \frac{d\omega(t)}{dt} \end{bmatrix} = \mathbf{A} \begin{bmatrix} i(t) \\ \theta(t) \\ w(t) \end{bmatrix} + \mathbf{B} \begin{bmatrix} v1(t) \\ v2(t) \\ v3(t) \end{bmatrix} \quad (3.77)$$

Como o valor da entrada é apenas da tensão controlada da bateria no sistema, v2(t) e v3(t) são iguais a zero.

O sistema será montado com base na equação acima, seguindo a seguinte ordem:

A matriz A será substituída pelos valores encontrados das 3 derivadas das variáveis de estado de forma que possam ser multiplicadas por seus correspondentes.

1º linha da matriz **A** e **B** = $\frac{-i(t)*Ra - Kb*\omega(t) + K_{ADC}*V_{bateria}(t)}{La}$

2º linha da matriz **A** = ω(t)

3º linha da matriz **A** = $\frac{Kt*ia(t) - b*w(t)}{Jt}$

Assim:

$$\begin{bmatrix} \frac{di(t)}{dt} \\ \frac{d\theta(t)}{dt} \\ \frac{d\omega(t)}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{Ra}{La} & 0 & -\frac{Kb}{La} \\ 0 & 0 & 1 \\ \frac{Kt}{Jt} & 0 & -\frac{b}{Jt} \end{bmatrix} * \begin{bmatrix} i(t) \\ \theta(t) \\ \omega(t) \end{bmatrix} + \begin{bmatrix} \frac{K_{ADC}}{La} \\ 0 \\ 0 \end{bmatrix} * \begin{bmatrix} V_{bateria}(t) \\ 0 \\ 0 \end{bmatrix} \quad (3.78)$$

Verificando na equação de estado $\dot{x}$, a primeira linha se refere a parte elétrica do sistema, e a 2º e 3º a parte mecânica do sistema.

B. Estado do Sistema em Relação a "y"

$$\begin{bmatrix} y1 \\ y2 \\ y3 \end{bmatrix} = \mathbf{C} \begin{bmatrix} i(t) \\ \theta(t) \\ \omega(t) \end{bmatrix} + \mathbf{D} \begin{bmatrix} v1(t) \\ v2(t) \\ v3(t) \end{bmatrix} \quad (3.79)$$

A partir daí, verifica-se para a matriz **C** qual relação com a saída das variáveis i(t), $\theta(t)$ e ω(t) se relacionam e este mesmo processo é realizado para a matriz de transmissão direta **D**.

No caso deste trabalho a saída deverá produzir o giro do eixo do motor (ω) e também calcular o ângulo deste giro(θ). Desta forma o sistema está relacionado com o ângulo e a

velocidade angular ($\theta(t) e\ \omega(t)$), ficando uma matriz de 3 linhas, mais usando somente 2. Assim:

1º Linha da matriz **C** = 0*i(t);

$$y1(t) = 0 \tag{3.80}$$

2º Linha da matriz **C** = K * $\theta(t)$;

Como existe um encoder acoplado ao motor CC, é ele que vai fazer a leitura do ângulo de giro do motor. Resultando:

$$y2(t) = K_{ENC} * \theta(t) \tag{3.81}$$

3º Linha da matriz **C** = K*ω(t).

Como existe um redutor acoplado ao motor CC, introduziremos ele na equação.

$$y3(t) = K_{RED} * \omega(t) \tag{3.82}$$

Substituindo as matrizes **C** e **D** na multiplicação por seus correspondentes, resulta:

$$\begin{bmatrix} y1 \\ y2 \\ y3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & K_{ENC} & 0 \\ 0 & 0 & K_{RED} \end{bmatrix} \begin{bmatrix} i(t) \\ \theta(t) \\ \omega(t) \end{bmatrix} + 0 \tag{3.83}$$

3.4.4 Resultado da equação de estado do sistema

$$\mathbf{A} = \begin{bmatrix} -\frac{Ra}{La} & 0 & -\frac{Kb}{La} \\ 0 & 0 & 1 \\ \frac{Kt}{Jt} & 0 & -\frac{b}{Jt} \end{bmatrix} \tag{3.84}$$

$$\mathbf{B} = \begin{bmatrix} \frac{K_{ADC}}{La} \\ 0 \\ 0 \end{bmatrix} \tag{3.85}$$

$$\mathbf{C} = \begin{bmatrix} 0 & K_{ENC} & 0 \\ 0 & 0 & K_{RED} \end{bmatrix} \tag{3.86}$$

$$\mathbf{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.87}$$

Para este caso, afiguram-se possíveis as seguintes saídas:

1º Linha da Matriz **C** = Saída relacionada ao Ângulo do rotor $\theta(t)$;

2º Linha da Matriz **C** = Saída relacionada a Velocidade Angular (ω(t)).

3.4.5 Montando a equação de estado de acordo com as características do sistema

A Tabela 3.3 apresenta as características do motor cc.

Tabela 3.3: Característica do motor CC de acordo com o fabricante

Parâmetro	Variável	Valor
Resistência da armadura	Ra	7,2
Indutância da armadura	La	3400µH
Constante de torque	Kt	13,15mN(m/a)
Constante da FCEM	Kb	1,38mV/rpm
Momento de inércia	Jt	4,5 g*cm^2
Atrito viscoso	b	Encontrar
Velocidade angular máxima	ω_{max}	9400rpm
Relação de transmissão	K_{RED}	1:84
Momento de inércia da carga	Jl	Encontrar
Atrito viscoso da carga	bl	Encontrar

Fonte: Fabricante do motor M223

De acordo com a Tabela 3.3, é necessário encontrar as unidades corretas para as variáveis Kb e Jt.

Transformar unidades

$$Kb = 13{,}178\text{e-}3\ \frac{\mathbf{V}}{\mathbf{rad/s}} \tag{3.88}$$

$$Jt = 4{,}5\text{e-}7\ K_g\text{m}^2 \tag{3.89}$$

Atrito viscoso

$$\omega max = = 984{,}36\ \text{rad/s} \tag{3.90}$$

$$b = \ 4{,}7989\text{e-}6\ \frac{\text{Nm}}{(\text{rad}^2/\text{s})} \tag{3.91}$$

Momento de inércia da carga

$$Iyy = 0{,}009375\ K_g m^2 \tag{3.92}$$

Atrito viscoso da carga

$$bl = 0{,}0075\text{N} \tag{3.93}$$

Para montagem da Equação de Estado, serão simuladas 2 situações, sendo a primeira o sistema sem carga e a segunda situação, o sistema com carga. Ainda para a primeira situação, será separada a simulação do sistema com redutor e do sistema sem redutor do motor.

Assim será demonstrado o equacionado sem carga de sistema A e o equacionado com carga de sistema B.

Sistema A – Sem Carga

Todo equacionamento desenvolvido até agora foi para o sistema A, segue abaixo o resultado.

$$\mathbf{A} = \begin{bmatrix} -\frac{Ra}{La} & 0 & -\frac{Kb}{La} \\ 0 & 0 & 1 \\ \frac{Kt}{Jt} & 0 & -\frac{b}{Jt} \end{bmatrix} = \begin{bmatrix} -2{,}1176e3 & 0 & -3{,}876 \\ 0 & 0 & 1 \\ 29{,}22e3 & 0 & -10{,}66 \end{bmatrix} \tag{3.94}$$

$$\mathbf{B} = \begin{bmatrix} \frac{K_{ADC}}{La} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1{,}4353 \\ 0 \\ 0 \end{bmatrix} \tag{3.95}$$

$$\mathbf{C} = \begin{bmatrix} 0 & K_{ENC} & 0 \\ 0 & 0 & K_{RED} \end{bmatrix} = \begin{bmatrix} 0 & 7{,}63 & 0 \\ 0 & 0 & \frac{1}{84} \end{bmatrix} \tag{3.96}$$

$$\mathbf{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.97}$$

Obs.: Para um sistema sem redutor a terceira linha da terceira coluna da matriz **C** será 1.

Sistema B – Com Carga

Para os sistemas com carga, altera-se somente a matriz **A** do equacionamento já realizado para o sistema sem carga, sendo que da matriz **A** será modificada somente a terceira linha que está relacionada com o momento de inércia "J" (terceira linha da primeira coluna) e com o atrito viscoso "b" (terceira linha da terceira coluna).

$$\boldsymbol{A} = \begin{bmatrix} -\frac{Ra}{La} & 0 & -\frac{Kb}{La} \\ 0 & 0 & 1 \\ \frac{Kt}{Jt+J1} & 0 & -\frac{b+bl}{Jt+J1} \end{bmatrix} = \begin{bmatrix} -2{,}1176e-3 & 0 & -3{,}876 \\ 0 & 0 & 1 \\ 1{,}4026 & 0 & -0{,}8 \end{bmatrix} \tag{3.98}$$

$$\frac{Kt}{Jt+J1} = \frac{13{,}15*10^{-3}}{\mathbf{4{,}5e-7+0{,}009375}} = 1{,}4026\ \frac{N}{K_g \mathrm{m}^2} \tag{3.99}$$

$$\frac{b+bl}{Jt+J1} = \frac{\mathbf{4{,}7989e-6+0{,}0075}}{\mathbf{4{,}5e-7+0{,}009375}} = 0{,}80 \tag{3.100}$$

Segue abaixo o sistema completo:

$$\mathbf{A} = \begin{bmatrix} -\frac{Ra}{La} & 0 & -\frac{Kb}{La} \\ 0 & 0 & 1 \\ \frac{Kt}{Jt+J1} & 0 & -\frac{b+bl}{Jt+J1} \end{bmatrix} \begin{bmatrix} -2{,}1176\text{e}-3 & 0 & -3{,}876 \\ 0 & 0 & 1 \\ 1{,}4026 & 0 & -0{,}8 \end{bmatrix} = \tag{3.101}$$

$$\mathbf{B} = \begin{bmatrix} \frac{K_{ADC}}{La} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1{,}4353 \\ 0 \\ 0 \end{bmatrix} \tag{3.102}$$

$$\mathbf{C} = \begin{bmatrix} 0 & K_{ENC} & 0 \\ 0 & 0 & K_{RED} \end{bmatrix} = \begin{bmatrix} 0 & 7{,}63 & 0 \\ 0 & 0 & \frac{1}{84} \end{bmatrix} \tag{3.103}$$

$$\mathbf{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.104}$$

3.4.6 Equação de estado – tempo discreto

Para que seja possível aplicar os resultados obtidos ao robô, é necessário transformar as informações do tempo contínuo para o tempo discreto. Assim foi utilizado um tempo de amostragem de 1Hz e foram utilizados o comando “c2d” do MatLab para transformar a equação obtida, chegando ao seguinte resultado:

Obs.: Como equação de referência para este trabalho, foi utilizado o sistema sem carga e com o redutor.

$$\mathbf{A} = \begin{bmatrix} -6{,}862\text{e}-31 & 0 & -4{,}818\text{e}-32 \\ 0{,}2151 & 1 & 0{,}01559 \\ 3{,}632\text{e}-28 & 0 & 2{,}55\text{e}-29 \end{bmatrix} \tag{3.105}$$

$$\mathbf{B} = \begin{bmatrix} 0{,}0001127 \\ 0{,}3039 \\ 0{,}3088 \end{bmatrix} \tag{3.106}$$

$$\mathbf{C} = \begin{bmatrix} 0 & 7{,}63 & 0 \\ 0 & 0 & \frac{1}{84} \end{bmatrix} \tag{3.107}$$

$$\mathbf{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.108}$$

3.5 Considerações do Capítulo

Pode-se observar a demonstração do modelamento cinemático e dinâmico para o robô, sendo o modelo cinemático representado através da cinemática direta e o modelo dinâmico representado no espaço de estado.

No próximo capítulo serão apresentados os resultados de simulações, tanto para a trajetória do robô, baseando-se no modelo cinemático, como para a aplicação de degrau e impulso no modelo dinâmico desenvolvido para verificar a estabilidade do sistema.

4 SIMULAÇÕES

4.1 Simulação da Trajetórias do Robô em Ambiente Matlab com base na Cinemática Direta.

Neste capítulo serão apresentadas as simulações sobre o modelo cinemático e dinâmico apresentados no capítulo anterior.

Tendo como referência o script trajetória desenvolvido por (MELO, 2007) para o modelo cinemático, foi elaborado um scripts para geração das trajetórias, baseando-se em sua velocidade angular conforme pode ser visto no item 4.3.

4.2 Script Trajetórias

Neste Script verifica-se no primeiro momento se as velocidades das rodas são iguais, se forem, utiliza-se a velocidade de uma das rodas para encontrar a posição de X e Y para cada segundo decorrido do deslocamento do robô. Caso contrário, calcula-se o raio de curvatura, a velocidade angular, o deslocamento angular para encontrar as posições X e Y, conforme é visto na Figura 4.1.

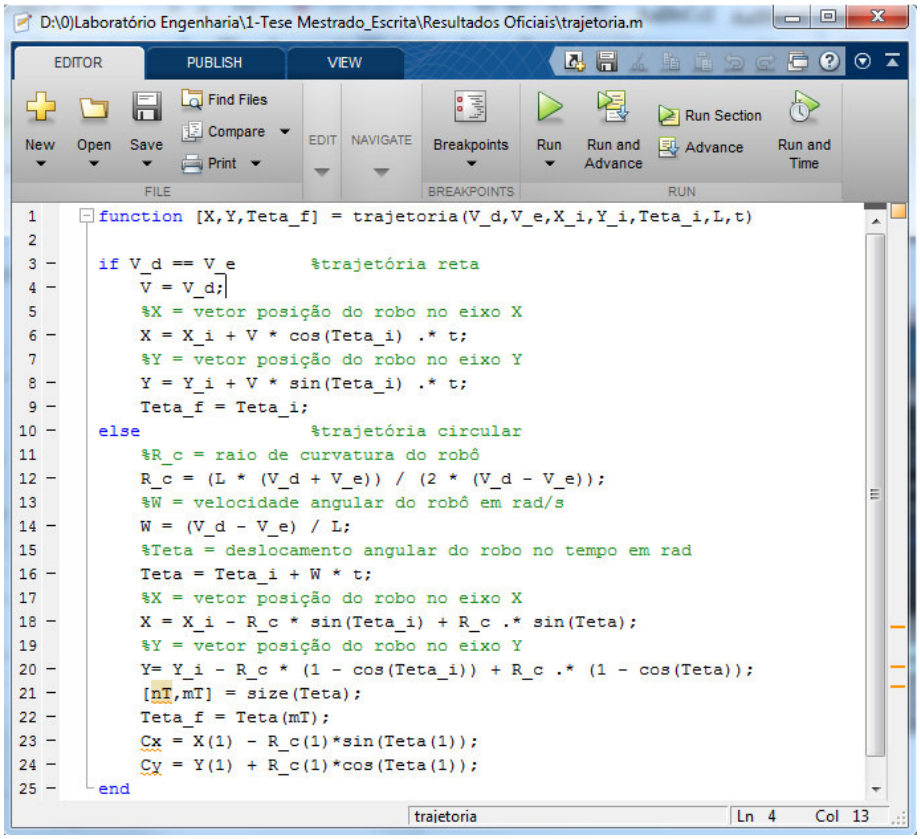


Figura 4.1: Script da trajetória

A lista de códigos da Figura 4.1 pode-se conferir no Apêndice C.

4.3 Script Geração das Trajetórias

Para gerar a trajetória, deve-se entrar com o valor da velocidade angular de cada roda do robô para Vrpm1, Vrpm1_2, etc, também informar a distância entre as rodas medidas em milímetros para L e o tempo da simulação, conforme pode ser visto na Figura 4.2.

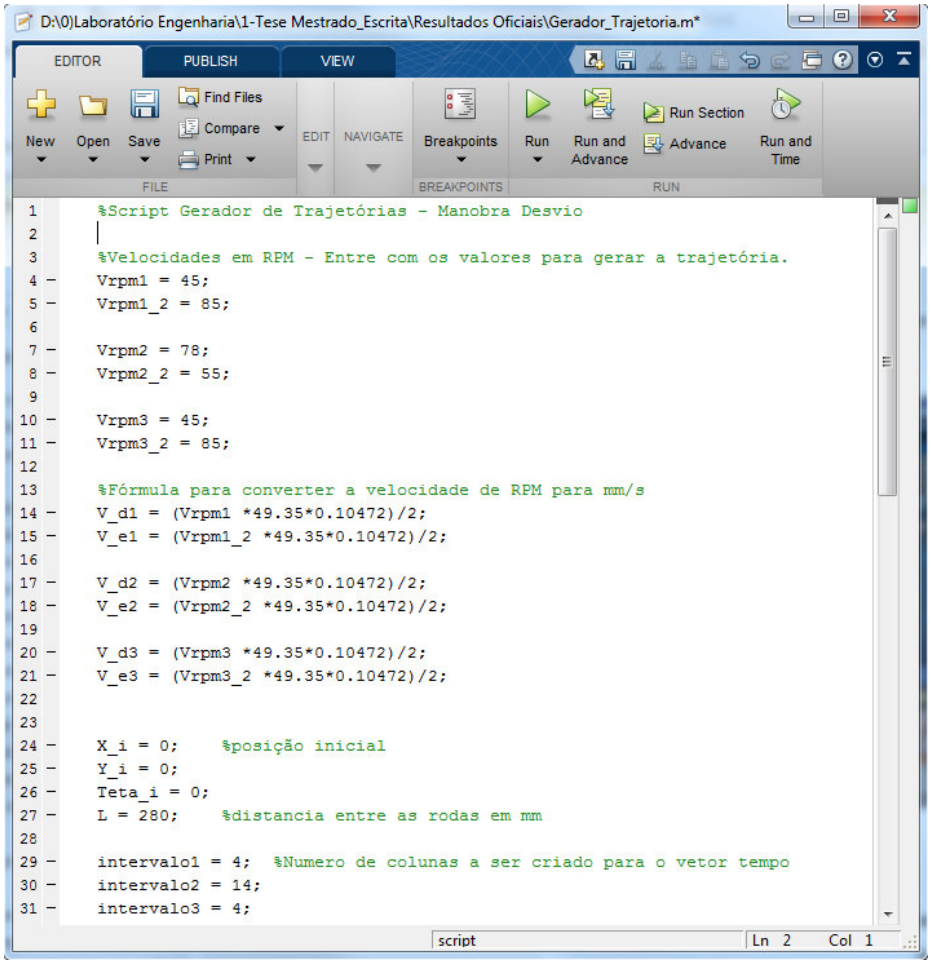


Figura 4.2: Script gerador de trajetórias

Na Figura 4.2 é apresentado a imagem do MatLab contendo o código gerador de trajetórias que também pode ser examinado por completo no Apêndice D.

4.3.1 Reta

Nesta manobra, como já foi explicado no conteúdo deste trabalho, a velocidade das duas rodas do robô devem ser iguais, diferindo somente na amplitude da velocidade, fazendo

com que o robô percorra uma trajetória mais rápido ou devagar de acordo com a velocidade aplicada.

A entrada dos valores de velocidade no simulador é referente a velocidade angular (ω). Pode-se transforma-la para a Velocidade linear tendo como unidade mm/s de acordo com a equação seguinte.

$$V = \frac{d*\omega*0.10472}{2} \tag{4.1}$$

Sendo ω a velocidade angular, V a velocidade linear das rodas de cada motor, medida em mm/s e "d" o diâmetro da roda.

- Para esta simulação são necessários os seguintes dados.
- ω_d = 21,1 RPM;
- ω_e = 21,1 RPM;
- Posição inicial sobre o eixo x,y e ângulo = 0;
- Distância entre as rodas = 280 mm.;
- Tempo de simulação = 19 s.

Sendo ω_d a velocidade angular da roda direita e ω_e a velocidade angular da roda esquerda.

A Figura 4.5 mostra a simulação da trajetória reta.

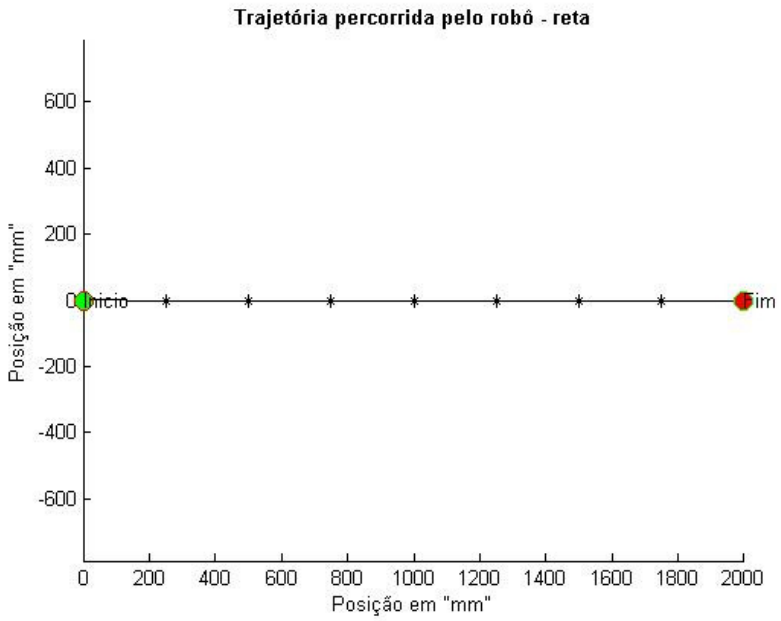


Figura 4.3: Simulação – Trajetória reta.

Utilizando a equação 4.1 é possível verificar que para uma roda percorrendo 54.52 mm a cada 1 segundo, sendo o tempo da simulação de 19 segundos, chega-se a uma distância de 54.52 x 19 resultando numa distância percorrida de 1035mm conforme é mostrado na Figura 4.5.

Obs.1: Cada quadrado em azul visualizado na figura acima equivale a 1s percorridos.

Obs.2: Para todas as simulações realizadas na sequência, serão aplicadas como posicionamento inicial do robô e ângulo de direção iguais a zero e distância entre as rodas de 280mm.

4.3.2 Circunferência com raio de 1 metro.

Para realização da Circunferência com raio de 1 metro são aplicadas velocidades fixas para cada roda do robô sendo suas velocidades diferentes. A roda que tiver a maior velocidade é que vai determinar o sentido da circunferência, sendo assim, se a roda direita tiver a maior velocidade o robô fará uma curva no sentido anti-horário, se for a roda esquerda, será no sentido horário.

Obs.: A referência de esquerda e direita do robô adotado nestas explicações é uma referência olhando o robô de trás para frente.

- ω_d = 81.4 RPM
- ω_e= 61.8 RPM
- Tempo Total = 34s;
- O tempo foi dividido em 4 pontos;
- Ponto Inicial = 0 s;
- Ponto 2 = 11 s;
- Ponto 3 = 11 s;
- Ponto Final = 12 s.

A Figura 4.6 mostra a simulação da trajetória circular de raio 1 metro.

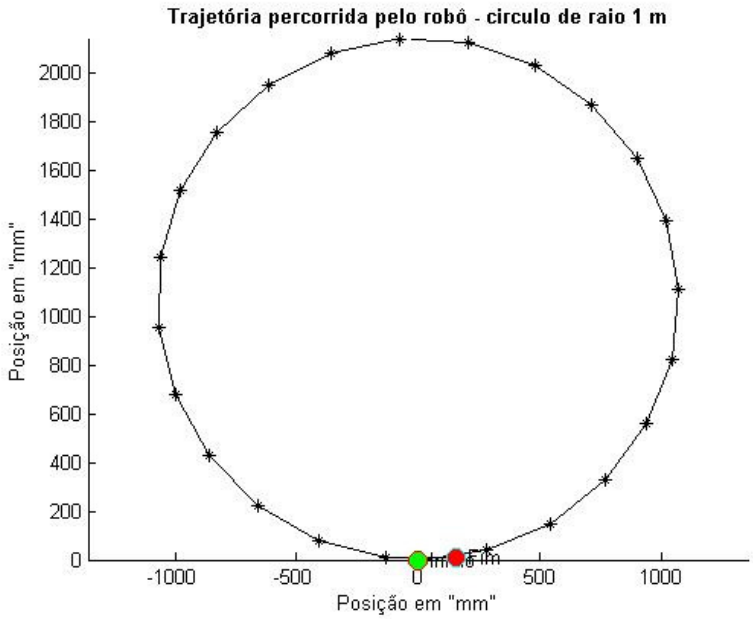


Figura 4.4: Simulação – Trajetória circular.

De acordo a Figura 4.6, verifica-se que o diametro da circunferência possui 2000 mm, percorrendo desta meneira uma circuferência com raio de 1000 mm.

4.3.3 Circular de raio 1 m.

De acordo com o desenvolvimento descrito no item 4.3.2 da circunferência de raio de 1 metro, agora será feito uma meia circunferência de raio de 1 metro de forma que o robô saia de seu ponto inicial desvie do objeto e posteriormente continue sua trajetória no mesmo sentido do ponto de origem.

A circunferência no tempo realiza-se da seguinte maneira:

Nos primeiros 4 segundos o robô anda com velocidade de 45 RPM na roda direita e 85 RPM na roda esquerda, após 4 segundos inverte-se as grandezas de valores das rodas direita e esquerda e mantidas durante mais 14 segundos. Finalmente após 14 segundos, é invertida novamente as grandezas de valores das velocidades das rodas direita e esquerda do robô durante mais 4 segundos para que o robô volte no sentido da trajetória inicial, conforme a Tabela 4.1.

Tempo Total = 22 s

Tempo foi dividido em 4 pontos

Tabela 4.1: Meia Circunferência

T(s)	T (s)	ω_d	ω_e
0	4	45	85
4	14	78	55
14	18	45	85
Fim		0	0

A Figura 4.5 ilustra a simulação da trajetória onde é percorrido meio círculo de raio 1 metro.

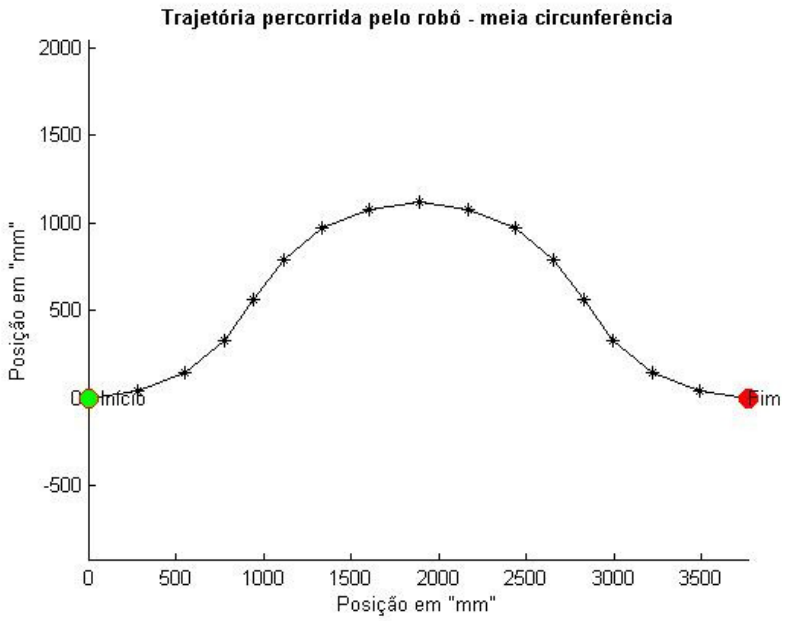


Figura 4.5: Simulação – Trajetória de meia circunferência.

No início o robô efetua uma curva no sentido anti-horário nos primeiros 4 segundos, já no 2º ponto o robô percorre uma curva no sentido horário com raio de aproximadamente 1 metro chegando a 18s, após no 3º ponto o robô realiza uma curva no sentido anti-horário chegando aos 22s e voltando à sua trajetória inicial.

4.4 Estabilidade do sistema sem feedback do encoder – Tempo Contínuo

Construída a equação de estado do sistema, verifica-se a estabilidade com base a resposta ao degrau e ao impulso que serão simulados no Matlab, nos moldes do equacionamento desenvolvido no capítulo 3 para as equações 3.102, 3.103, 3.104, 3.105.

A. Sistema sem Carga

i) Sem redutor ($K_{RED} = 1$)

A Figura 4.6 ilustra o lugar das raízes para o ângulo.

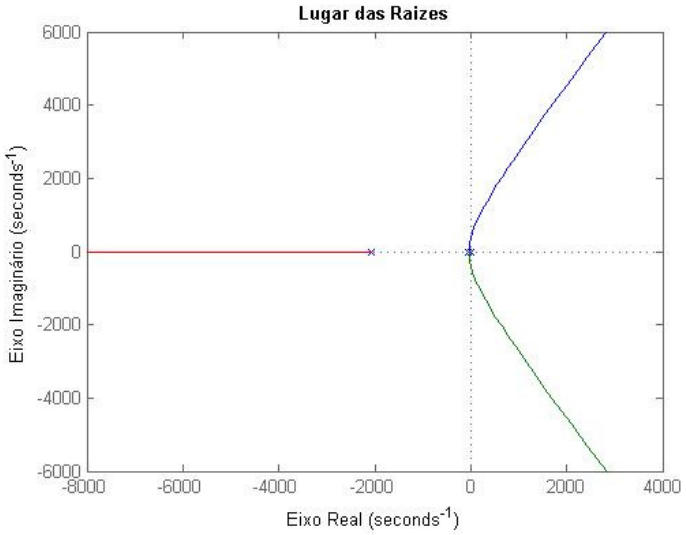


Figura 4.6: Gráfico do lugar das raizes para o ângulo.

De acordo com a localização dos polos do angulo na Figura 4.6, observa-se que existe polos no eixo imaginário, tornando este sistema instável.

A Figura 4.7 ilustra o lugar das raízes para a velocidade.

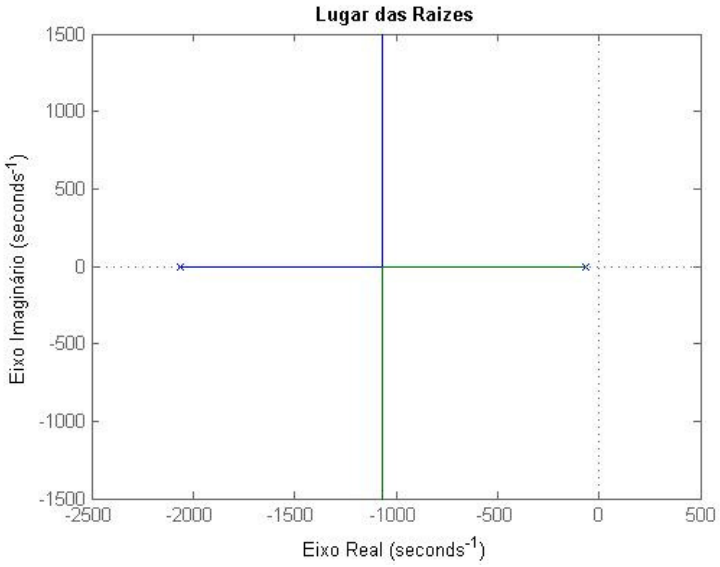


Figura 4.7: Gráfico do lugar das raizes para a velocidade.

Na Figura 4.7 os polos da velocidade estão do lado esquerdo do eixo imaginário sendo assim o sistema estável.

Resposta ao Degrau

A resposta ao degrau pode ser vista na Figura 4.8.

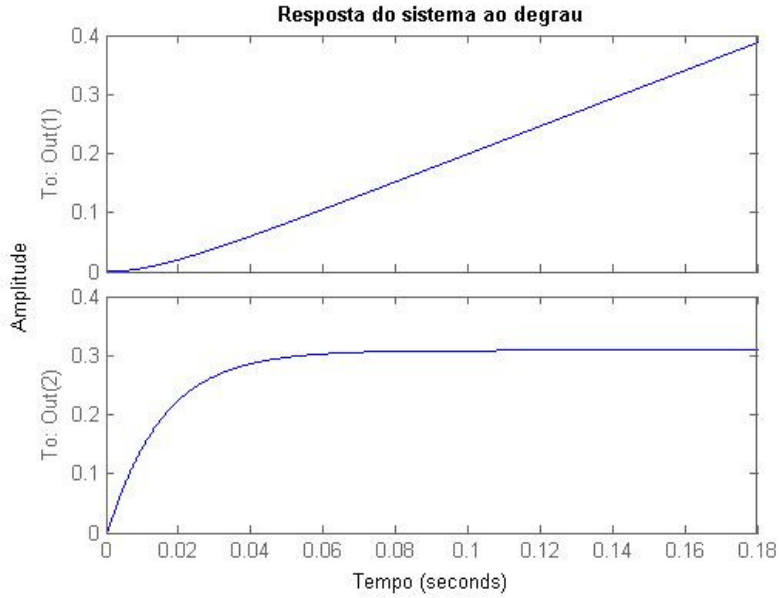


Figura 4.8: Resposta do sistema ao degrau.

Na Figura 4.8 mostra-se dois gráficos, sendo o primeiro gráfico a resposta do sistema para o ângulo e o segundo a resposta do sistema para a velocidade. Este mesmo processo é verificado na Figura 4.9.

Resposta ao Impulso

A Figura 4.9 ilustra a resposta do sistema ao impulso.

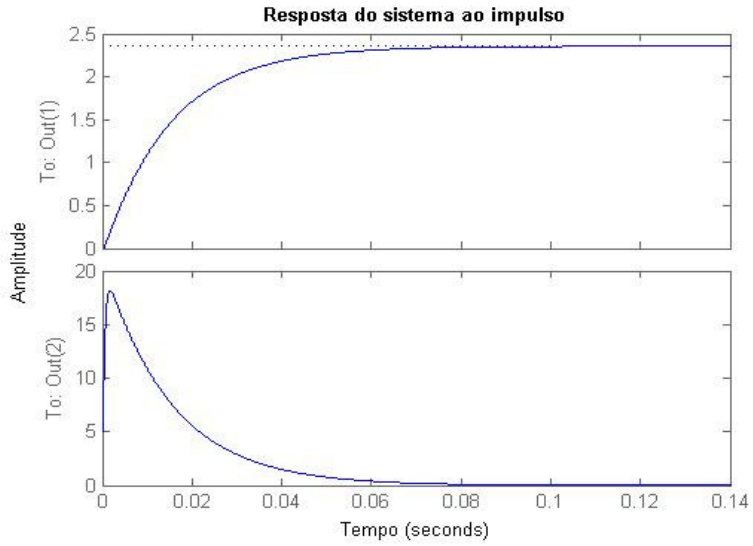


Figura 4.9: Resposta do sistema ao impulso.

Figura acima ilustra a resposta do sistema ao impulso.

Conforme a matriz **C** que foi descrita 3.4.6, os gráficos da Figura 4.8 e da Figura 4.9 apresenta estabilidade somente para o controle da velocidade do sistema, já que em relação ao ângulo não houve estabilidade nem para resposta ao degrau e nem para o impulso. Isto também pode ser observado conforme a posição dos polos apresentados na Figura 4.6 e Figura 4.7.

ii) Com redutor (K_{RED} = 1/84)

A Figura 4.10 ilustra o lugar das raízes para o ângulo.

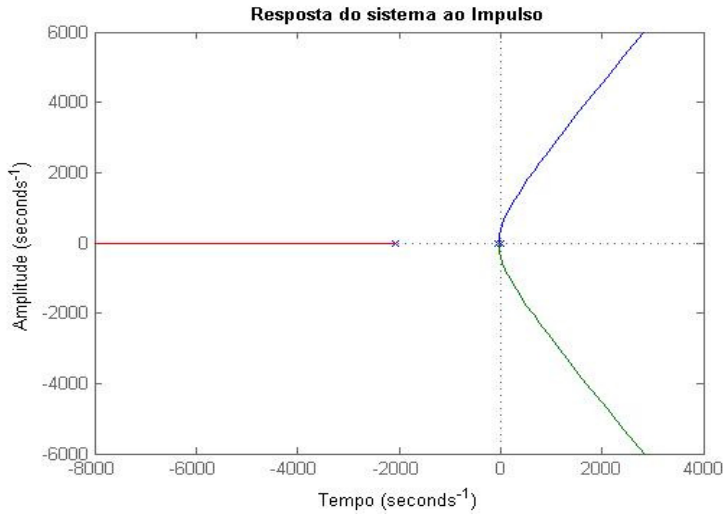


Figura 4.10: Gráfico do lugar das raizes para o ângulo – com redutor.

Neste gráfico verifica-se a instabilidade do sistema por conta da localização dos polos do sistema.

A Figura 4.11 ilustra o lugar das raízes para a velocidade

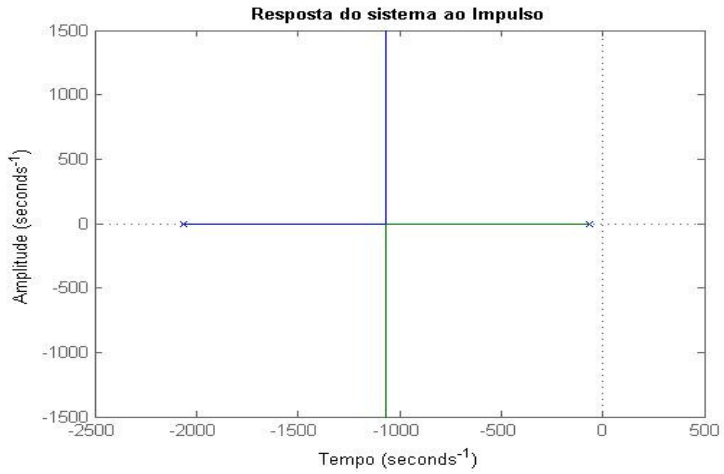


Figura 4.11: Gráfico do lugar das raizes para a velocidade – com redutor.

A figura Figura 4.11 mostra a posição dos polos do sistema, apura-se um sistema estável.

Resposta ao Degrau

A Figura 4.12 ilustra a resposta do sistema ao degrau para o ângulo e velocidade, sistema com redutor.

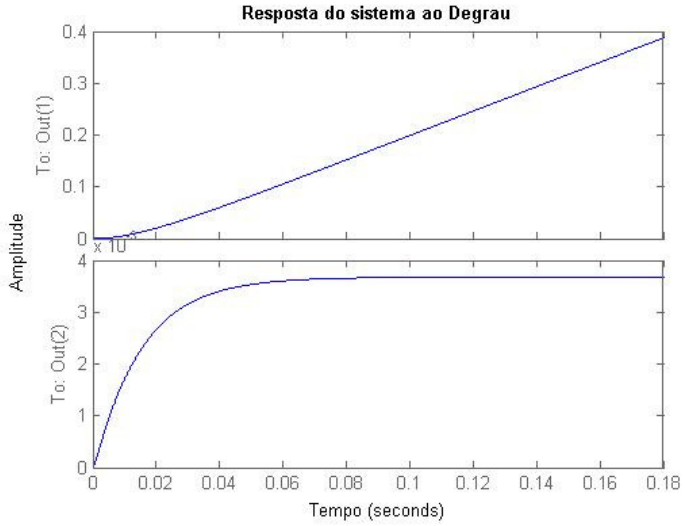


Figura 4.12: Resposta do sistema ao degrau – com redutor.

A Figura 4.12 ilustra a resposta do sistema ao degrau para o ângulo e velocidade respectivamente, sendo utilizado neste sistema o redutor em conjunto com o motor.

Resposta ao Impulso

A Figura 4.13 ilustra a resposta do sistema ao impulso, para o ângulo e velocidade, sistema com redutor.

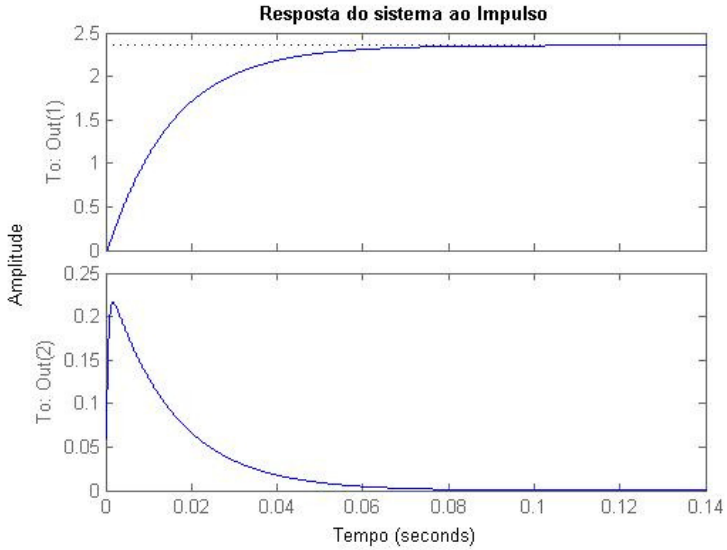


Figura 4.13: Resposta do sistema ao impulso – com redutor.

Veja-se que, com a inclusão do redutor no sistema, nota-se de acordo com a Figura 4.12 e Figura 4.13 houve-se alteração somente na resposta da velocidade, tendo uma reposta igual porém com diferença na amplitude, sendo agora menor. Confere-se que a inclusão do redutor não altera em nada a resposta do sistema em relação ao ângulo.

B. Sistema com Carga

i. Sem redutor ($K_{RED} = 1$)

A Figura 4.14 ilustra o lugar das raízes para o ângulo.

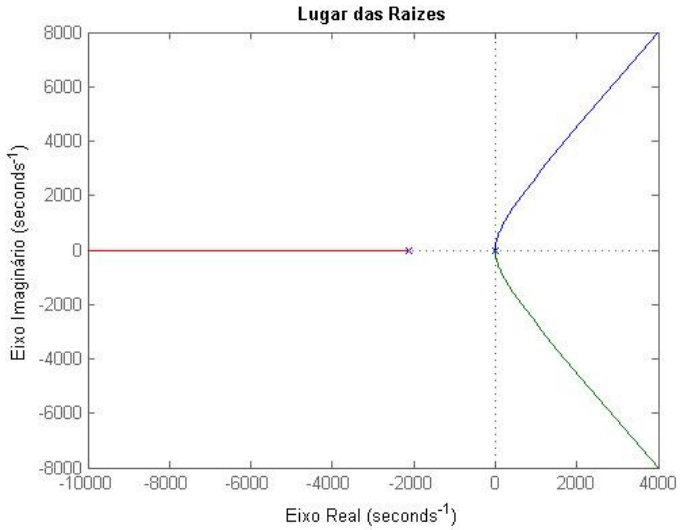


Figura 4.14: Gráfico do lugar das raizes para o ângulo – com carga.

Este gráfico mostra a posição dos polos para o ângulo, confere-se a instabilidade do sistema conforme a localização dos polos do sistema.

A Figura 4.15 ilustra o lugar das raízes para a velocidade.

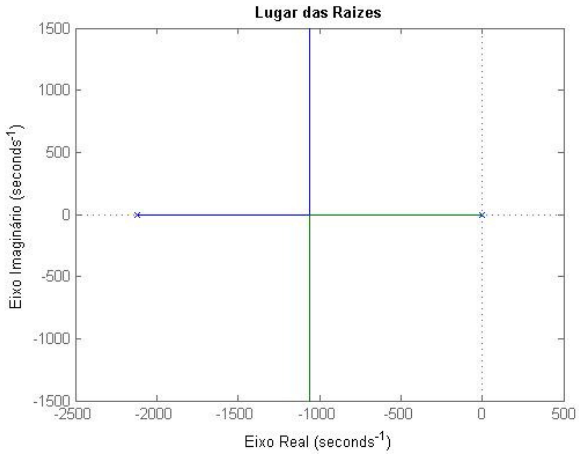


Figura 4.15: Gráfico do lugar das raizes para a velocidade – com carga.

No gráfico da Figura 4.15 é possível identificar a estabilidade do sistema para a velocidade conforme a localização dos polos.

Resposta o Degrau

A Figura 4.16 ilustra a resposta do sistema ao degrau, sistema com carga.

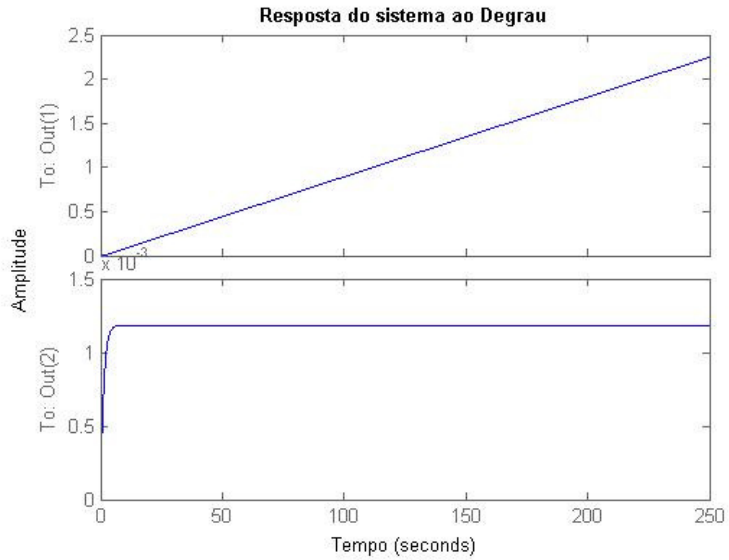


Figura 4.16: Resposta do sistema ao degrau – com carga.

Conforme o gráfico ilustrado na Figura 4.16 identifica-se a resposta do sistema para o ângulo no gráfico superior e para a velocidade no gráfico inferior, sendo esta resposta igual a Figura 4.12, porém com a amplitude menor para a velocidade devido a utilização da carga.

Resposta ao Impulso

A Figura 4.17 ilustra a resposta do sistema ao impulso.

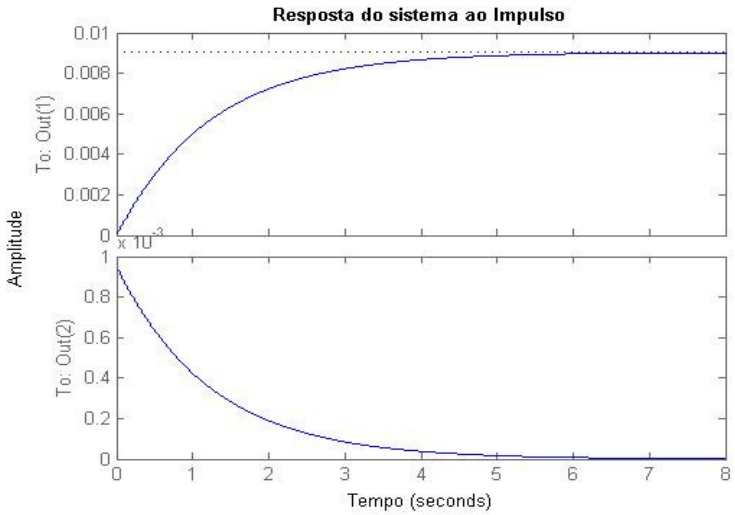


Figura 4.17: Resposta do sistema ao impulso – com carga.

De acordo com a Figura 4.16 e a Figura 4.17 uma resposta semelhante ao sistema com redutor porém de amplitude menor para a velocidade e com o aumento do tempo de resposta para o ângulo e velocidade devido ao uso da carga junto ao sistema.

ii. Com redutor (K_{RED} = 1/84)

A Figura 4.18 ilustra o lugar das raízes para o ângulo.

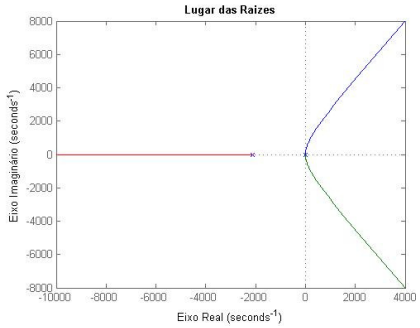


Figura 4.18: Gráfico do lugar das raizes para o ângulo – com carga e redutor.

Neste gráfico mostra-se o lugar das raízes para o ângulo. Apura-se a instabilidade do sistema por conta disto.

A Figura 4.19 ilustra o lugar das raízes para a velocidade.

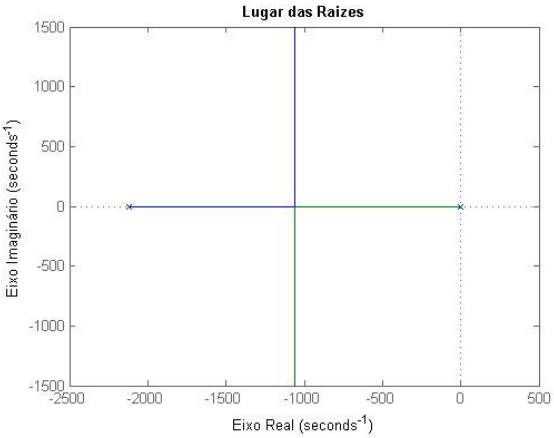


Figura 4.19: Gráfico do lugar das raizes para a velocidade – com carga e redutor.

Diferente da resposta da Figura 4.18, para o gráfico da Figura 4.19 certifica-se a estabilidade do sistema de acordo com o posicionamento dos polos.

Resposta ao Degrau

A Figura 4.20 ilustra a resposta do sistema ao degrau.

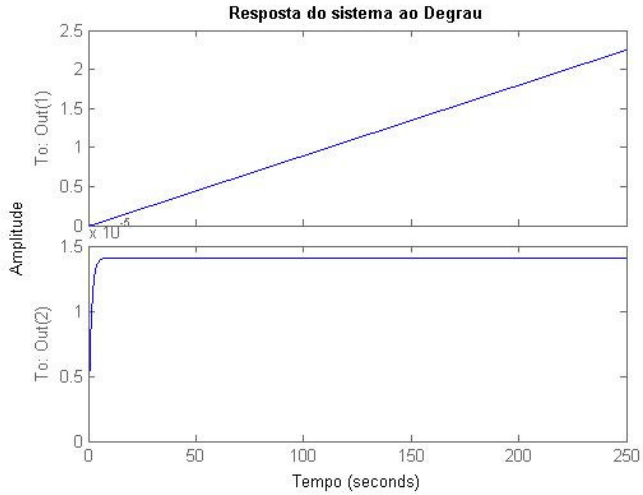


Figura 4.20: Resposta do sistema ao degrau – com carga e redutor.

Este sistema tem resposta equivalente a Figura 4.16, diferenciando-se somente na amplitude para a velocidade que se encontra no gráfico inferior da Figura 4.16.

Resposta ao Impulso

A Figura 4.21 ilustra a resposta do sistema ao impulso.

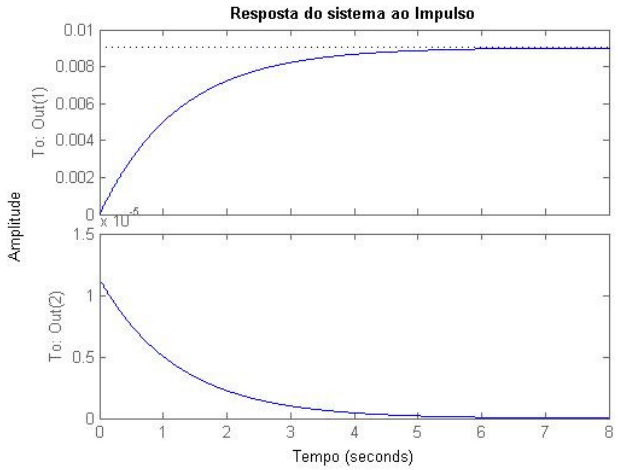


Figura 4.21: Resposta do sistema ao impulso – com carga e redutor.

Para o sistema acima representado, a entrada do redutor fica semelhante as outras respostas já analisadas aqui, ou seja, não interfere na resposta do ângulo e diminui a amplitude da velocidade angular.

4.5 Estabilidade do sistema sem feedback do encoder– Tempo Discreto

A finalidade desta simulação é verificar se o método utilizado para discretizar o sistema é correto. As simulações serão implementadas para o caso que iremos utilizar no robô, que é o sistema sem carga e com redutor.

A. Sistema sem Carga

ii) Com redutor (K_{RED} = 1/84)

Resposta ao Degrau

A Figura 4.22 ilustra a resposta do sistema ao degrau.

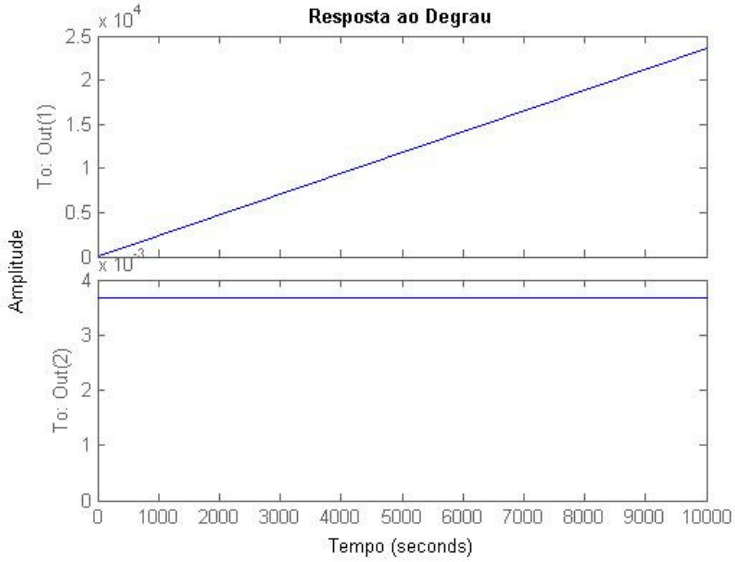


Figura 4.22: Resposta do sistema ao degrau – Discreto.

A Figura 4.22 ilustra-se a resposta do sistema ao degrau, para um sistema discreto, onde no gráfico superior se encontra a resposta para o ângulo e no gráfico inferior a resposta para a velocidade.

Resposta ao Impulso

A Figura 4.23 ilustra a resposta do sistema ao impulso.

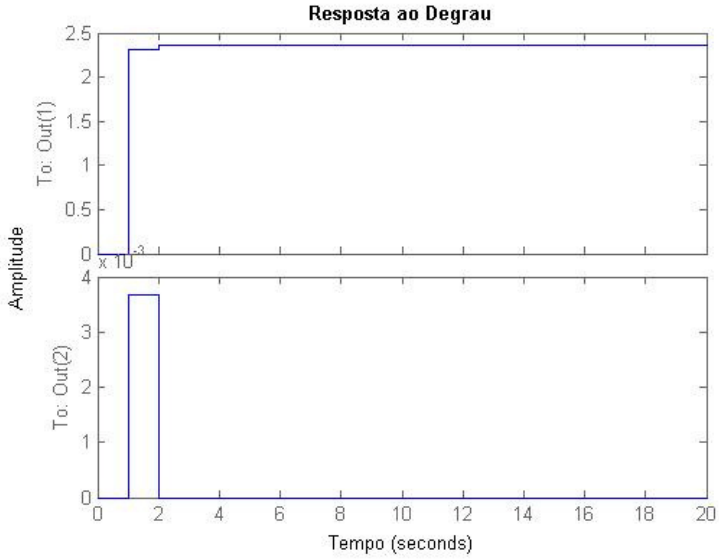


Figura 4.23: Resposta do sistema ao impulso – Discreto.

Como a resposta ao degrau para a velocidade e impulso para o angulo da Figura 4.22 e da Figura 4.23 é igual ao tempo contínuo conforme pode ser vista na Figura 4.12 e Figura 4.13, apura-se que o método de discretização utilizado é válido. Logo, esta função deve ser aplicada ao implementar os testes práticos no robô.

4.6 Estabilidade do sistema sem feedback do encoder– Controle PI

No ponto em análise será implementado um controlador para a Velocidade Angular (ω), a fim de que a resposta do sistema seja igual ou próximo ao sinal de entrada.

Após a comprovação do modelo matemático do sistema e análise da estabilidade do sistema para a velocidade angular, passa-se para o controle do sistema em sem o feedback do encoder com controle proporcional e integral para a velocidade angular. Desta forma será realizada a análise do gráfico da resposta da velocidade angular ao degrau, conforme a Figura 4.24:

A Figura 4.24 ilustra a resposta do sistema ao degrau, aplicado para análise do sistema para implementar o controlador.

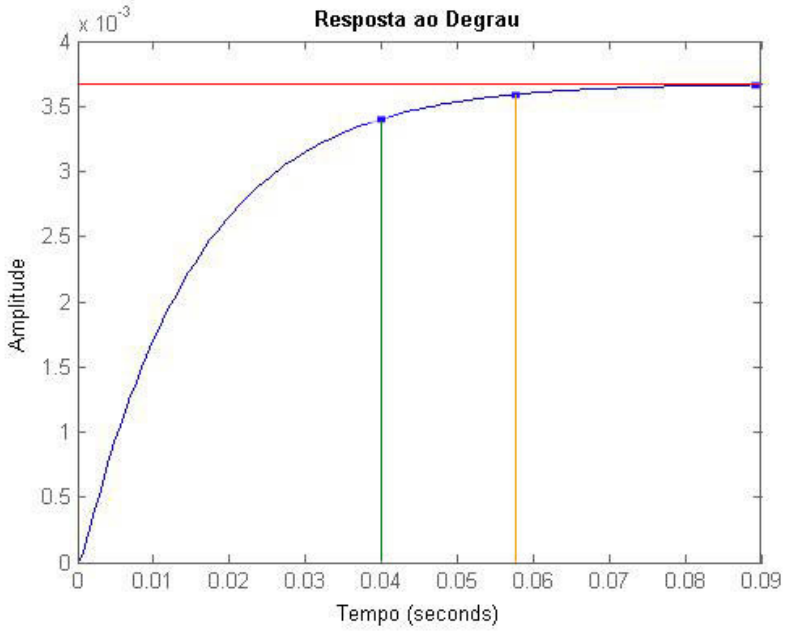


Figura 4.24: Análise do sistema para implementação do controlador.

- Linha Vermelha = valor final da resposta;
- Linha Verde = tempo de subida;

- Linha Laranja = tempo de estabilização do sistema.

Pode-se verificar que o valor final da resposta de acordo com a linha em vermelho, é 3,66*10e-3, logo o ganho DC da função é de 1/273,2. Isto corresponde a um erro de estado estacionário de:

$$Erro = 1 - 3{,}66*10e\text{-}3 \tag{4.2}$$

Onde 1 é o valor do degrau aplicado e 3,66*10e-3 é o valor da resposta obtida do sistema.

$$Erro = 0{,}99634 \tag{4.3}$$

O valor do Erro é muito alto, e deve ser corrigido pelo controle PI que será encontrado. Também de acordo com a Figura 4.24, o tempo de subida é de 0,04 s, e o tempo de estabilização é de cerca de 0,058 s.

Os valores de Kp e Ki serão encontrados diretamente no MatLab utilizando o comando "*pidtool*".

4.6.1 Controle proporcional

Com o objetivo de reduzir o tempo de subida e principalmente reduzir o erro de estado estacionário, será utilizado o Controle Proporcional. Assim, conforme descrito acima, por meio da ferramenta do MatLab obtém-se a resposta ao degrau.

Para facilitar trabalhar com a ferramenta "pidtool" do matlab, será transformada a equação de estado do sistema, em uma função de transferência, tendo como saída somente a velocidade angular. Para fazer esta transformação, emprega-se o comando "*ss2tf*" do matlab. Segue abaixo o resultado relacionado ao sistema sem carga e com redutor.

$$FT = \frac{499{,}3s}{s^3 + 2128\, s^2 + 135800\, s} \tag{4.4}$$

Desta forma, encontra-se o seguinte ganho:

$$K_p = 415{,}2265 \tag{4.5}$$

Como resultado tem-se a seguinte equação:

$$FT = \frac{207s}{s^3 + 2128\, s^2 + 135800\, s} \tag{4.6}$$

Obtendo a seguinte resposta ao degrau do sistema mostrado na Figura 4.25.

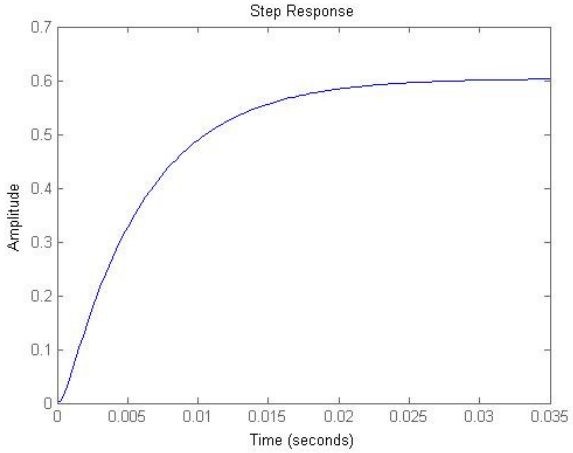


Figura 4.25: Resposta ao Degrau para velocidade - controlador P.

- *Tempo de Subida* = 0,0126s
- *Tempo de Estabilização* = 0,0228s

Contrapondo o gráfico da Figura 4.24 com o gráfico da Figura 4.25, apura-se que houve um bom resultado no sistema de controle, já que diminui enormemente o erro de estado estacionário, o tempo de subida e o tempo de estabilização. Sabendo que o Controle Proporcional não é capaz de eliminar por completo o erro de estado estacionário, adota-se um Integrador em conjunto.

4.6.2 Controle proporcional + integral

Neste controle, busca-se reduzir o erro estacionário tentando não aumentar a resposta transitória, encontrando-se o seguinte ganho:

$$K_p = 149{,}7285 \tag{4.7}$$

$$K_i = 29376{,}6426 \tag{4.8}$$

Como resultado tem-se a seguinte equação:

$$FT = \frac{74400\ s^2 + 14.670.000.000\ s}{s^4 + 2128\ s^3 + 210200\ s^2 + 14.670.000.0000\ s} \tag{4.9}$$

Obtendo a seguinte resposta ao degrau do sistema mostrado na Figura 4.26.

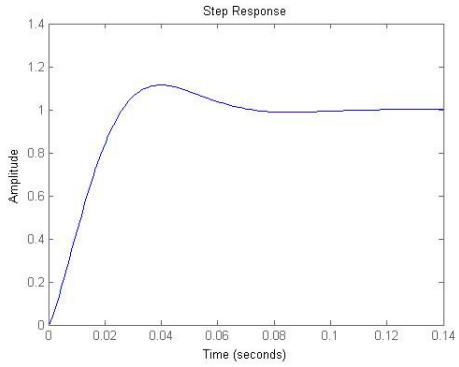


Figura 4.26: Resposta do sistema para Velocidade – controlador PI .

- *Tempo de Subida* = 0,019s
- *Tempo de Estabilização* = 0,0645s
- *Overshoot* = 11.4%

Fazendo a comparação do gráfico da Figura 4.26 em relação ao gráfico da Figura 4.25, confere-se que houve um aumento no tempo de subida em 0,00640 s e também houve uma elevação no tempo de Estabilização de 0,0417 s, além do overshoot agora existente, porém o valor do erro de estado estacionário se extinguiu, podendo considerar este sistema satisfatório, porquanto aqui os fatores positivos são muito mais compensadores que os fatores negativos.

Tempo de Subida = 0,0126 s

Tempo de Estabilização = 0,0228 s

4.7 Estabilidade do sistema com feedback do encoder– Controle PI

O Controle do sistema sem o feedback do encoder foi eficiente, consoante demonstrado no item 4.3. Porém agora é necessário fazer o controle com o feedback do encoder para que, quando o robô estiver se movendo no solo, seja facilmente controlado com os dados do encoder, evitando que reduza a velocidade e que se auto ajuste com a diferença de seus motores.

Assim, encontra-se a seguinte equação para o sistema:

$$FT = \frac{499{,}3s}{s^3 + 2128\, s^2 + 136300\, s} \tag{4.10}$$

Na sequência, segue-se o valor do ganho proporcional e integrador do sistema (PI).

$$K_p = 149{,}8811 \tag{4.11}$$

$$K_i = 29532{,}3607 \tag{4.12}$$

Feito isso, encontra-se o seguinte resultado:

$$FT = \frac{7{,}484e4\ s^2 + 1{,}475e7\ s}{s^4 + 2128\ s^3 + 2{,}107e5\ s^2 + 1{,}475e7\ s} \tag{4.13}$$

A Figura 4.27 ilustra a resposta do sistema ao degrau.

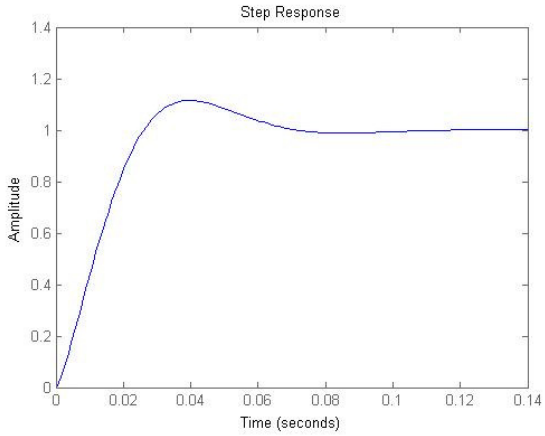


Figura 4.27: Resposta do sistema para Velocidade – PI – MF.

- *Tempo de Subida* = 0,0189s
- *Tempo de Estabilização* = 0,0643s
- *Overshoot* = 11.4%

Na prática, como já descrito acima, será possível utilizar a realimentação para ler os valores do encoder e conseguir controlar o sinal de saída para que fique o mais próximo do sinal de entrada.

4.8 Modelo Final – Digital

De acordo com todas as simulações feitas, foi encontrado nesta última simulação, realizada para um sistema com feedback do encoder e com controle PI, o modelo final para um sistema sem carga e com redutor, que será aplicado ao robô. Desta forma, fazendo a discretização deste modelo, para um frequência de amostragem de 1Hz, tem-se a equação 4.14.

$$FT = \frac{z^2 - 1{,}67e{-}22\ z - 4{,}7488e{-}46}{z^3 - 3{,}2806e{-}22\ z^2 + 2{,}7528e{-}44\ z - 1{,}4012e{-}82} \quad (4.14)$$

Na Figura 4.28 segue a resposta ao degrau e na sequência impulso deste sistema.

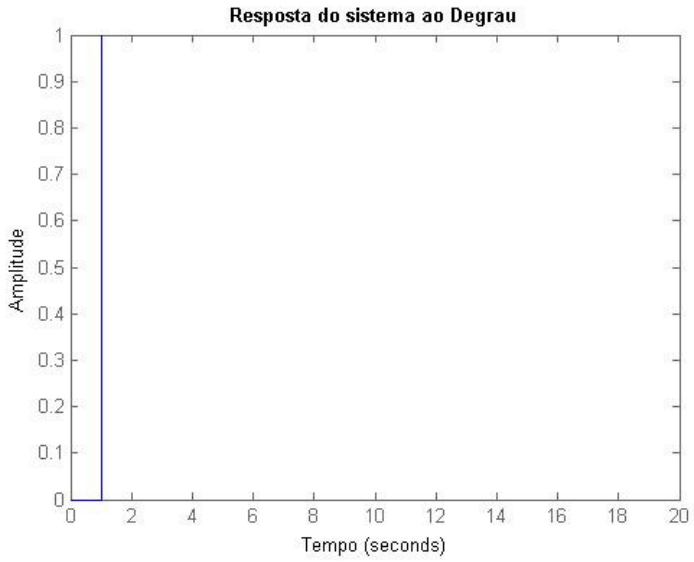


Figura 4.28: Resposta ao degrau para velocidade – PI – Discreto.

O que se vê desta representação é a resposta do sistema ao degrau, sistema discreto com controlador PI.

A Figura 4.29 ilustra a resposta do sistema ao impulso discreto.

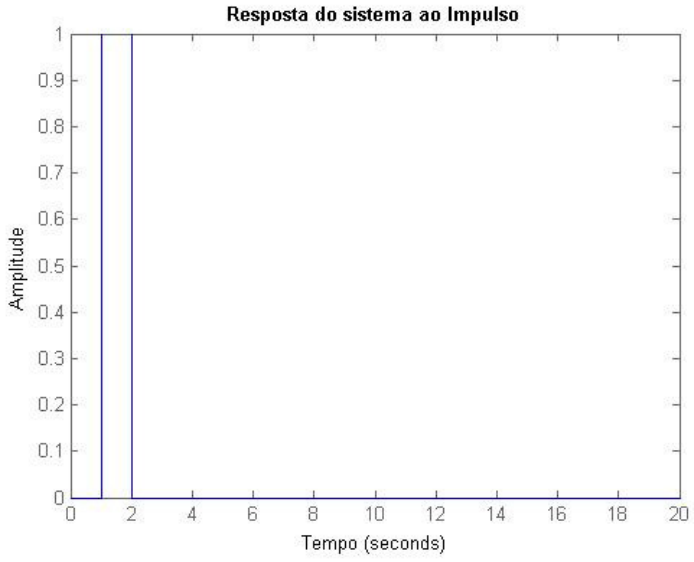


Figura 4.29: Resposta ao impulso para velocidade – PI – Discreto.

Assim, nos termos da resposta mostrada na Figura 4.29, comprova-se que função de transferência discreta apresenta um resultado correto.

4.9 Considerações do Capítulo

Foram apresentados neste capítulo as simulações sobre o modelo cinemático em relação às três trajetórias desenvolvidas, reta, circunferência de raio de um metro e meia circunferência de raio de um metro, além das simulações de estabilidade sobre o modelo dinâmico, verificando que somente a velocidade angular utilizando um controle PI está apta a controlar a velocidade do robô.

5 IMPLEMENTAÇÕES E RESULTADOS EXPERIMENTAIS

Neste capítulo será apresentada a implementação dos resultados experimentais desenvolvidos para um sistema sem e com o feedback do encoder.

Os Testes a se realizar neste capítulo estão relacionados com a comunicação que será feita entre Computador(PC) e o microcontrolador ATmega 2560 através da porta USB/serial.

O Kit Arduino Mega 2560 tem um toolbox para ser utilizado no MatLAb para sua programação em tempo real com técnicas de hardware-in-the-loop. De acordo com (MELLO, 2007) esta técnica de simulação é utilizada nos sistemas de prototipagem rápida para controladores robóticos moveis embarcado, onde pode ser utilizada para o desenvolvimento e parametrização de controladores de sistemas embarcados. As simulações de hardware-in-the-loop proveem uma plataforma efetiva de desenvolvimento por adicionarem a complexidade da planta sob controle à plataforma de teste. O sistema de controle é incluso nos testes e desenvolvimentos através de seus modelos de representações matemáticas e todos os respectivos modelamentos dinâmicos (ABRÓSIO, 2005).

A simulação de hardware-in-the-loop inclui a emulação elétrica de sensores e atuadores que atuam com interface entre a simulação da planta e o sistema embarcado que está sendo testado. O valor de cada sensor emulado eletricamente é controlado pelo simulador da planta e é utilizado pelo sistema embarcado. Desta forma, o sistema embarcado implementa os algoritmos de controle agindo nos atuadores do sistema (PAGNOTELLI e VALIGI, 2006).

Para conseguir programar o algoritmo de controle com maior poder de processamento, será utilizado o trabalho conjunto MatLab-Simulink onde o Computador(PC) irá interagir com os componentes conectados a placa Arduino Mega 2560, aplicando para isto as portas de saídas e entradas do microcontrolador. Assim é possível escrever e executar os programas diretamente a partir do MaTLab ou Simulink.

Para conseguir testar a estabilidade do robô de acordo com as equações desenvolvidas no capítulo 3, foi necessário instalar um compilador da linguagem de programação “C” no Simulink e mais um pacote de suporte do MatLab para o Arduino (ArduinoIO Package) para permitir que a comunicação do sistema de simulação MatLab e Simulink com as portas de entradas e saídas do Microcontrolador. Este suporte MatLab para Arduino permite a

comunicação com uma placa arduino através de uma porta serial. Ele consiste de uma API MatLab no computador Host e um programa de servidor que roda no Arduino. Este servidor fica rodando na placa, que escuta os comandos que chegam via porta serial, executa os comandos, e, se necessário, retorna um resultado (mathworks, 2014).

5.1 Estabilidade do sistema sem feedback do encoder

Para o Caso Prático, é fundamental simular o sistema contendo apenas 1 saída, vez que cada saída da simulação está ligada a 1 saída do microcontrolador. O sistema em malha aberta é usado para os casos onde não existe o encoder para dar o feedback do sinal de entrada na saída. Desta forma, não sendo possível verificar a velocidade angular do robô naquele momento, faz-se o controle em malha aberta ou seja, sem o feedback do encoder.

Todos as simulações feitas na prática são referentes ao sistema com redutor e sem carga.

5.1.1 Resposta ao degrau.

Resposta ao Degrau – Velocidade e Angulo

Degrau indo de 0 a 1

A Figura 5.1 ilustra o diagrama de blocos da simulação realizada no Simulink. O sistema não tem o feedback do encoder.

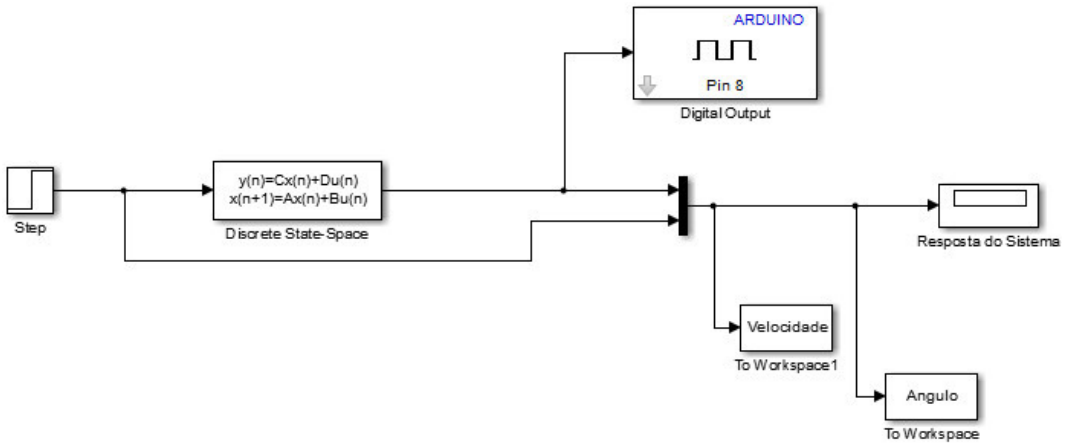


Figura 5.1: Simulador da reposta ao degrau.

A Figura 5.1 representa um sistema de controle em sem feedback do encoder, para um sistema sem carga e com redutor, igual ao item 4.2-A-ii. Este sistema está diretamente ligado ao robô, e após a compilação verifica-se o resultado no scopo “Resposta do Sistema”.

Para selecionar a saída desejada configura-se apenas uma saída na matriz **C**. Caso se pretenda averiguar a saída para o ângulo, seleciona-se a matriz para o ângulo, o que também se aplica para encontrar a velocidade. Deve-se compilar o sistema para cada saída desejada.

Desta forma utiliza-se um bloco para obter os resultados discretos do script montado para gerar a equação de estado discreta no MatLab, que conforme a Figura 5.1, após aplicar um degrau neste sistema, obtém o resultado que será enviado para o microcontrolador através do bloco Digital Output, neste caso para o pino 8 do microcontrolador, onde foi ligado um dos motores. Na Figura 5.2 apresenta-se o Script utilizado para a montagem da equação de estado discreto.

A Figura 5.2 ilustra o Script gerador da equação de estado discreto.

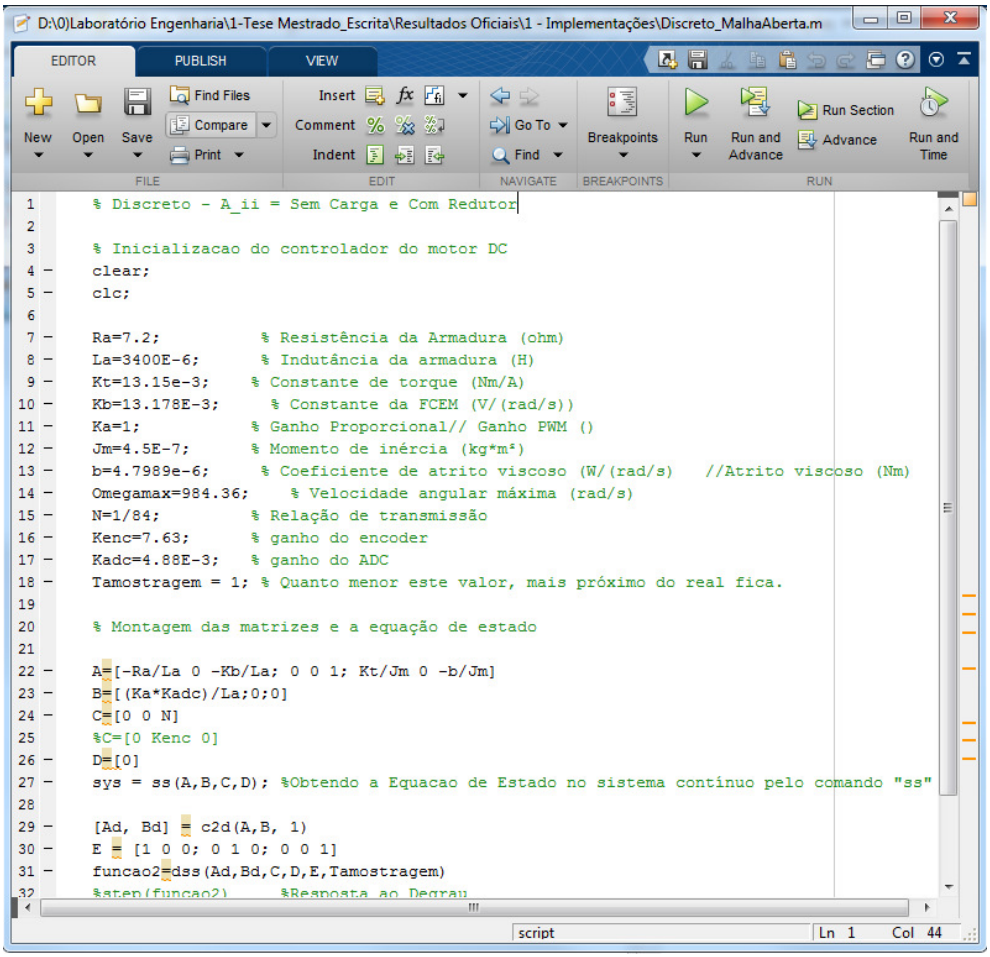


Figura 5.2: Script gerador da equação de estado discreto.

Assim, rodando o script da Figura 5.2, é gerada a equação de estado do sistema discreto, que será usada pelo programa criado no simulink. De tal modo os resultados são mostrados na Figura 5.3 e Figura 5.4.

- Resposta ao Degrau – Velocidade

A Figura 5.3 ilustra a resposta ao degrau para velocidade.

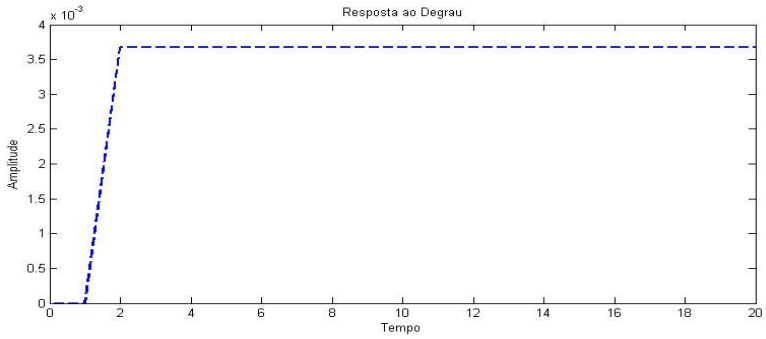


Figura 5.3: Reposta ao degrau – velocidade zoom.

De acordo com o degrau unitário aplicado neste sistema, mostra-se na Figura 5.3 a resposta da velocidade angular, coincidindo com o sistema simulado no capítulo 4 conforme a Figura 4.12.

- Resposta ao Degrau – Ângulo do Rotor

A Figura 5.4 ilustra a resposta ao degrau para ângulo.

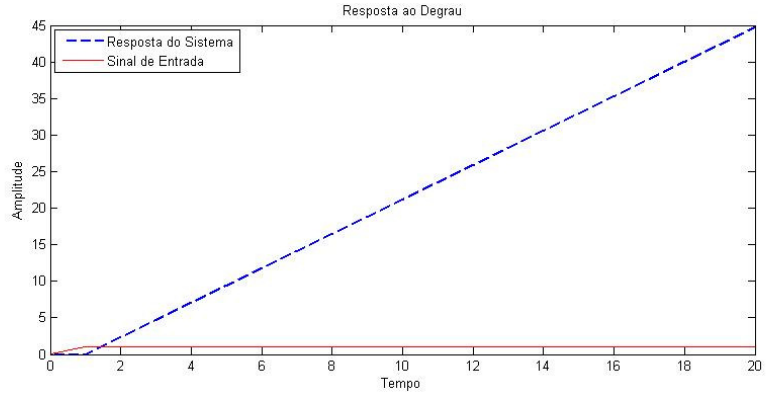


Figura 5.4: Reposta ao degrau – ângulo.

Na Figura 5.4 a linha contínua é o degrau unitário aplicado no sistema e a linha pontilhada a resposta do sistema para o ângulo, verificando-se a mesma resposta simulada conforme a Figura 4.12.

Obs.: Ao aplicar esta equação ao robô, o motor ligou, e pouco tempo depois se desligou por causa da instabilidade.

5.1.2 Resposta Ao Impulso

Para aplicação da resposta ao impulso, será utilizado o simulador da resposta ao impulso conforme a ilustra a Figura 5.5.

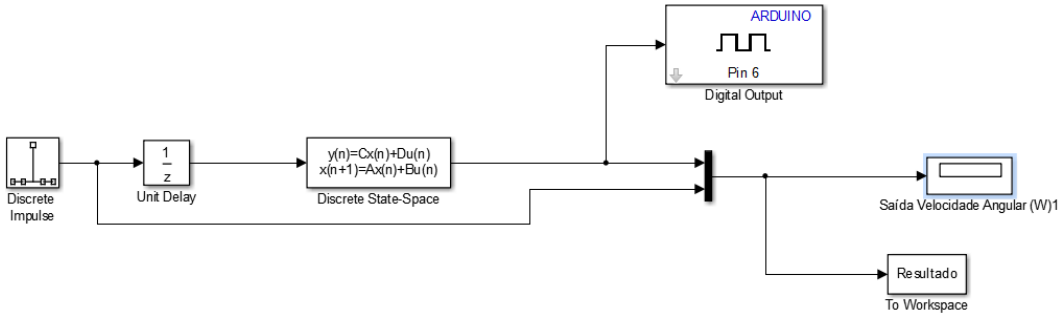


Figura 5.5: Simulador da reposta ao impulso.

- Resposta ao Impulso – Velocidade Angular

A Figura 5.6 ilustra a resposta ao impulso para a velocidade.

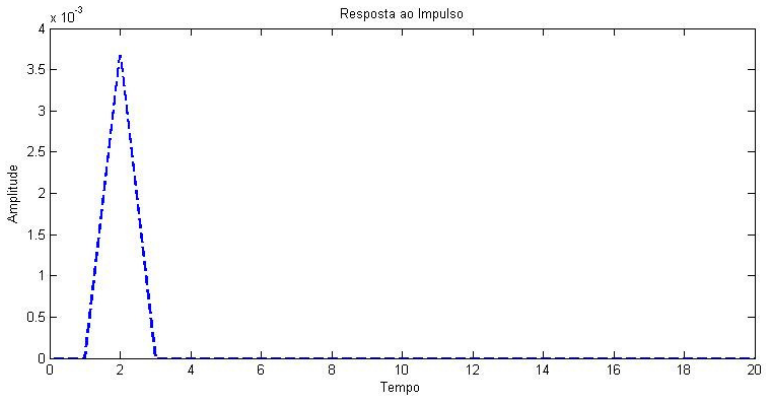


Figura 5.6: Reposta ao impulso – velocidade.

Foi gerado um impulso de amplitude 1 verificando que a resposta assemelha-se com aquela do modelo teórico para um tempo discreto, conforme pode ser visto na Figura 4.13.

Resposta ao Impulso – Ângulo do Rotor

A Figura 5.7 ilustra a resposta ao impulso para o ângulo.

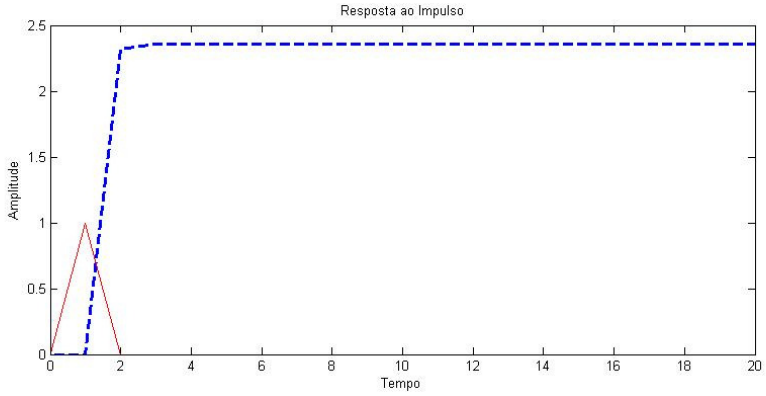


Figura 5.7: Reposta ao impulso – ângulo.

Para este teste, verifica-se a instabilidade do controle do sistema pelo ângulo, sendo esta resposta coincidente com o sistema simulado, conforme pode ser verificado na Figura 4.13.

Nota-se que mesmo após o impulso ter sido finalizado, o motor desliga-se em poucos segundos, sendo não instável para controle pelo ângulo do rotor.

5.2 Estabilidade do Sistema com Feedback do Encoder – Controle PI

Este é o modelo final que será testado e que servirá de referência para o controle do robô. O modelo já foi desenvolvido na seção 4.8, todavia, neste momento, será implementado na prática. A Figura 5.8 apresenta o diagrama de blocos simulado no Simulink.

5.2.1 Resposta ao Degrau

Para este modelo, serão simulados apenas as respostas ao degrau e impulso para a velocidade angular, que é o sistema que será trabalhado, conforme é ilustrado na Figura 5.8.

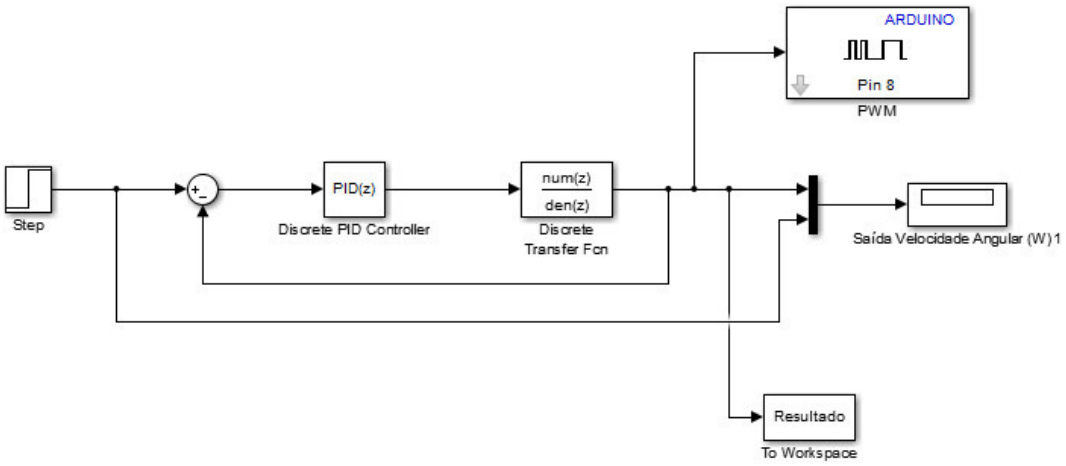


Figura 5.8: Simulador da reposta ao degrau – PI.

Esta construção é uma sequência do que já foi visto nas seções 5.1.1e 5.1.2, sendo agora o sistema com feedback do encoder, utilizando o bloco de controle discreto PID, onde só foi usado o controle PI, conforme pode ser visto na Figura 5.8.

Conforme já explicado na seção 5.1.1, é utilizado o script apresentado na Figura 5.9 para a formação do sistema, porém neste caso, foi utilizando a equação de transferência, conforme já explicado na seção 4.6.1.

A Figura 5.9 ilustra o script gerador da equação de transferência discreto.

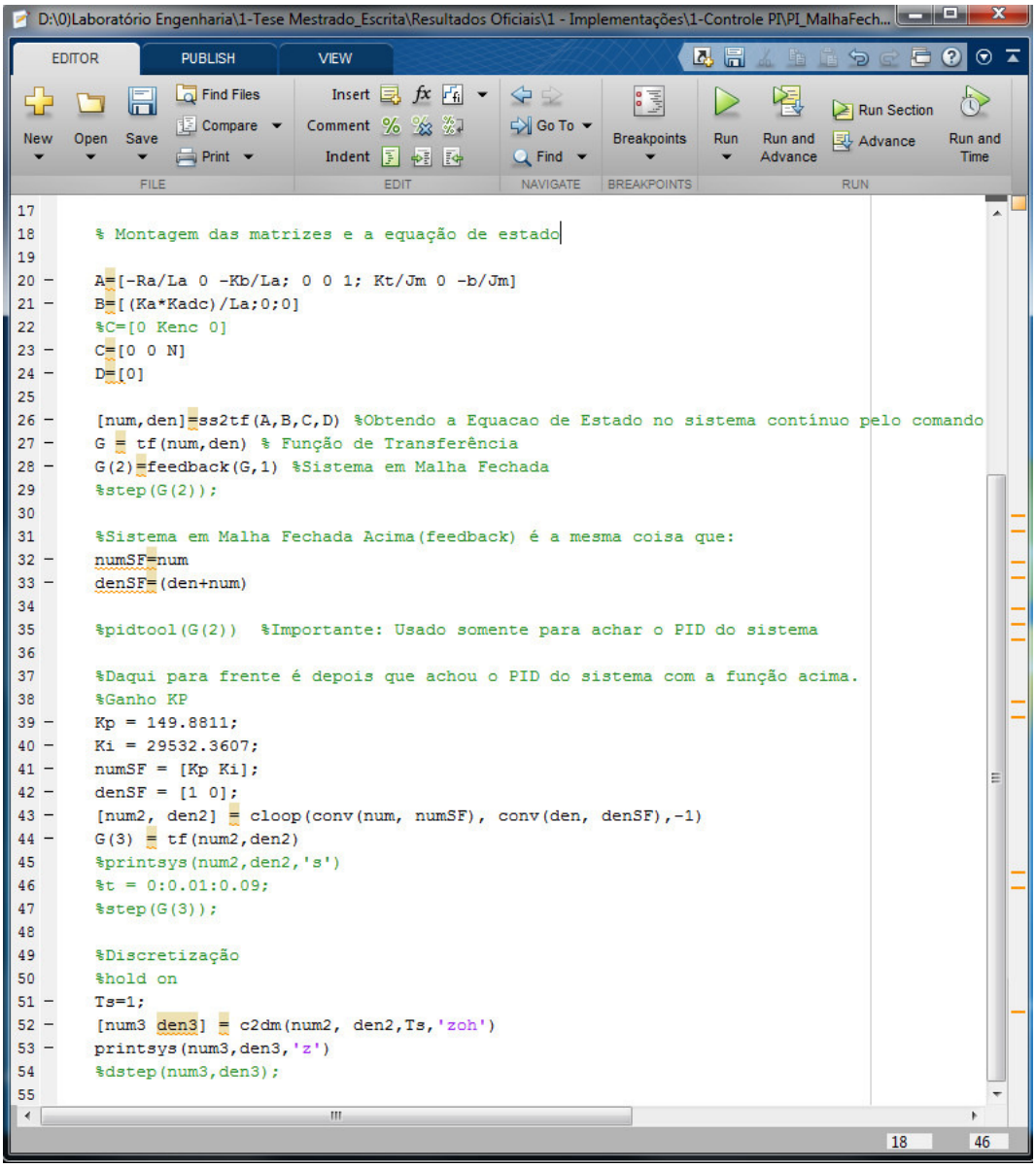


Figura 5.9: Script gerador da equação de transferência discreto.

O código deste script pode ser conferido no Apêndice B . Rodando o script será gerada a função de transferência do sistema em com o feedback do encoder, para um sistema sem carga e com redutor. A Figura 5.10 ilustra a resposta do sistema.

- Resposta ao Degrau – Velocidade

A Figura 5.10 ilustra a resposta do sistema ao degrau.

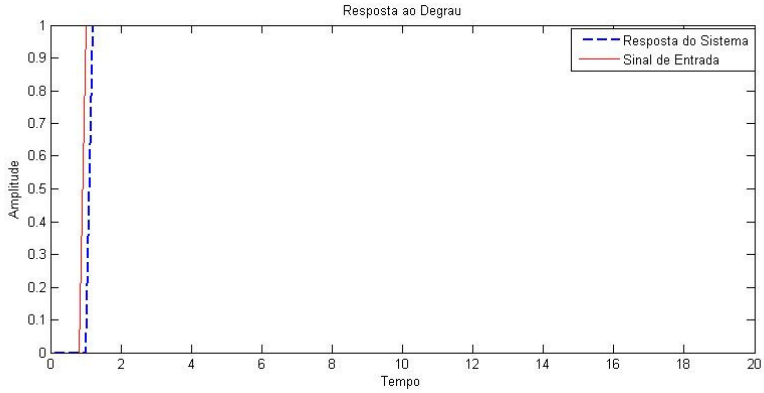


Figura 5.10: Resposta ao degrau – PI – Velocidade.

Na Figura 5.10, a linha contínua à esquerda é o degrau aplicado ao sistema e a linha tracejada à direita é a reposta do sistema a entrada do sinal. Verifica-se a estabilidade do sistema para a resposta ao degrau para a velocidade, sendo esta resposta coincidente com o sistema simulado no capítulo 4, o que pode ser visto na Figura 4.28..

5.2.2 Resposta ao impulso

Na Figura 5.11 ilustra o sistema construído para testar a resposta ao impulso para um sistema com controle PI.

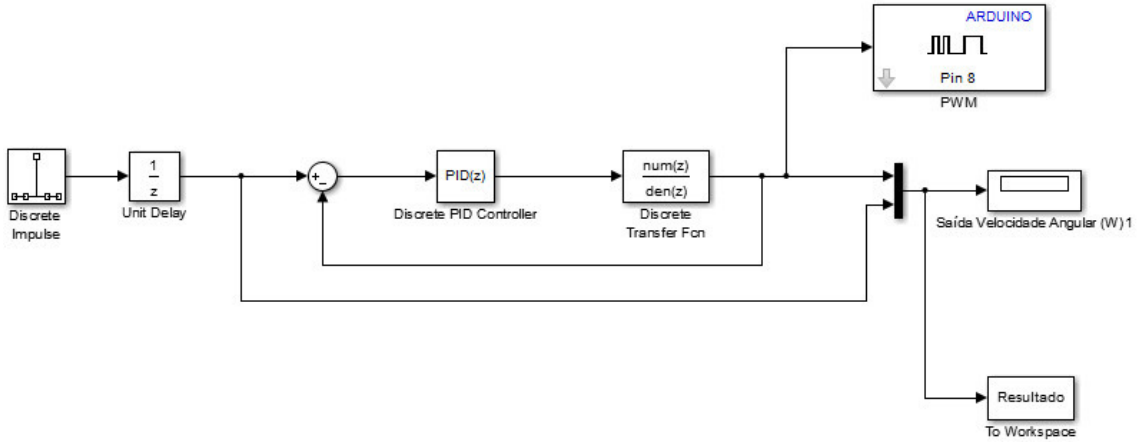


Figura 5.11: Simulador da reposta ao impulso – PI.

- Resposta ao Impulso – Velocidade

A Figura 5.12 ilustra a resposta do sistema ao impulso.

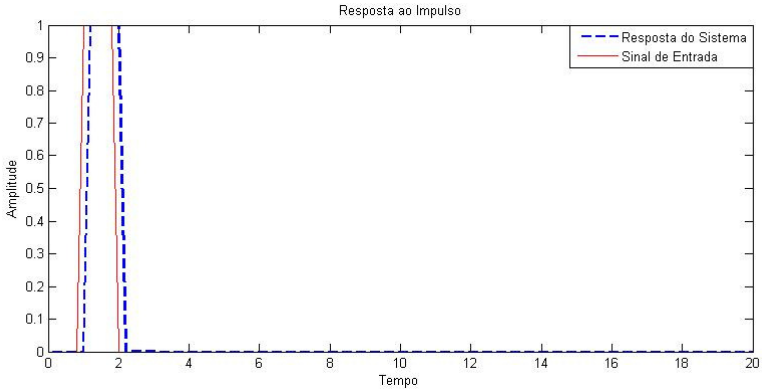


Figura 5.12: Resposta ao Impulso – PI – Velocidade.

Nesta figura, a linha contínua é a entrada do sinal e a linha pontilhada é a resposta do sistema. Para a resposta ao impulso, o sistema continua estável em relação à velocidade, sendo coincidente com o sistema simulado conforme pode verifica-se na Figura 4.29.

5.3 Implementação do Simulador Virtual – Sem o feedback do encoder

Neste item será apresentado o controlador do robô. Através do Simulink foi elaborado um controlador baseado na teoria desenvolvida e simulada da cinemática direta e dinâmica, tendo como objetivo enfatizar o controle das trajetórias do robô. Na Figura 5.13 exibe-se a visão geral do sistema sem o feedback do encoder.

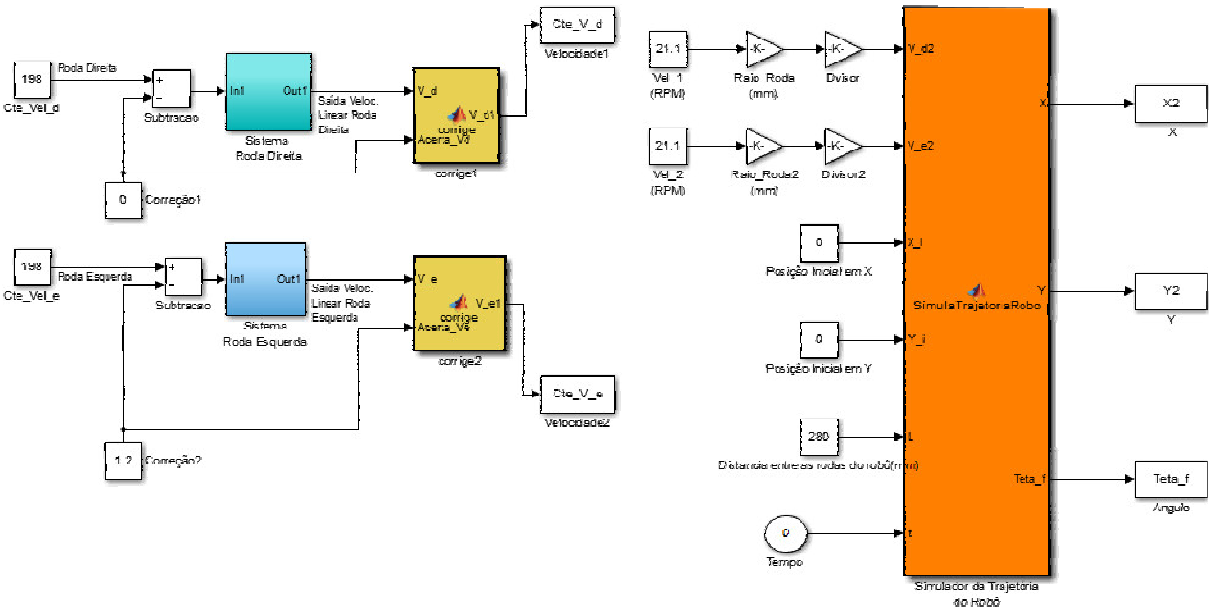


Figura 5.13: Simulador virtual – MA.

Para este sistema, foram construídos quatro tipos de bloco, separados por entrada, controle de motor CC, correção de velocidade e cinemática direta. Serão explicadas cada uma delas, separadas pelo seu tipo.

5.3.1 Bloco de entrada

A Figura 5.14 ilustra o bloco de entrada.

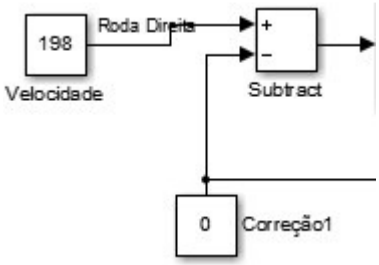


Figura 5.14: Bloco de entrada.

Os valores das entradas dos blocos podem variar de 0 a 255 (microcontrolador de 8 bits). Para controlar as velocidades dos motores CC, conforme já explicado na seção 2.5.5, a velocidade varia de 0 a 100% do Duty Cycle, porém, para simulação realizada no Matlab essa relação é invertida.

Portanto, quando se insere o valor de 0 no bloco de entrada da Figura 5.14, será o valor máximo (100% Duty Cycle) na saída do PWM do microcontrolador, e se entrar com o valor de 255 será o valor mínimo (0% Duty Cycle) na saída PWM do microcontrolador.

Junto com os blocos de entrada se encontram os blocos de correção, pois de acordo com os teste realizados no robô, verificou-se que mesmo os motores sendo de igual modelo, existe uma diferença de velocidade para a mesma entrada um em relação ao outro, motivo pelo qual elaborou-se junto à simulação a opção de fazer a correção manualmente.

5.3.2 Bloco de controle dos motores

Na sequência iniciam-se as seções de controle, onde situa-se o bloco de controle dos motores. A estrutura de blocos é mostrada na Figura 5.15.

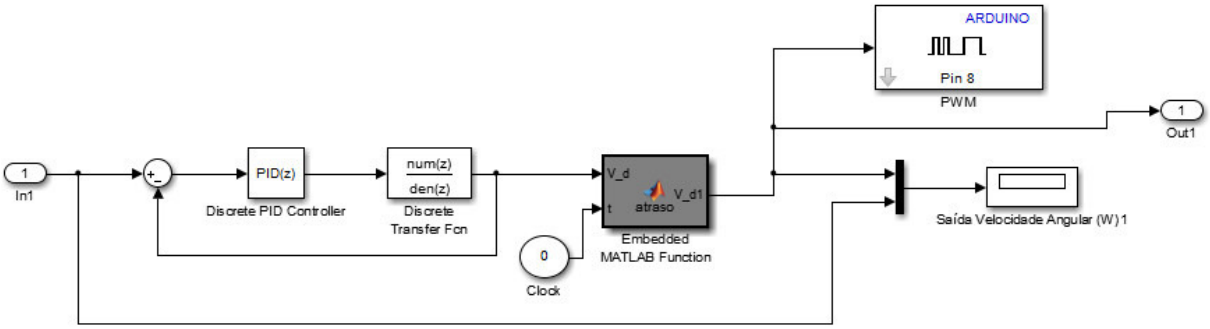


Figura 5.15: Bloco de controle dos motores.

Este sistema é o mesmo sistema já explicado na seção 4.4, tendo como acréscimo o bloco de controle de tempo. Devido aos valores serem invertidos da relação Simulink e microcontrolador, durante a inicialização do sistema o valor inicial é igual a zero, o que faz o motor inicializar em sua velocidade máxima. O bloco, então, controla o valor inicial do motor, que para um tempo menor 1 segundo, o valor da velocidade fica sendo igual a 255 correspondente a 0% Duty Cycle do microcontrolador.

É neste bloco também que configura-se o tempo que se deve para o motor, já que na prática, após ser compilado o código, se não houver esta configuração o motor permanece ligado infinitamente executando o algoritmo.

5.3.3 Bloco de correção de velocidade

Bloco de correção de velocidade final é ilustrado na Figura 5.16.

Figura 5.16: Bloco de correção de velocidade.

Este bloco tem como objetivo deixar a velocidade equilibrada em relação às correções que foram feitas na entrada das diferenças entre os dois motores, para não haja qualquer alteração na simulação da trajetória que será vista posteriormente.

5.3.4 Bloco da trajetória

O bloco trajetória, apresentado na Figura 5.17 possui variáveis de entrada V_d2, V_e2, X_i, Y_i, Teta_i, L, t e saídas X, Y, Teta_f e tem por finalidade verificar se as entradas de sinais estão correspondendo a trajetória que deve ser percorrida pelo robô.

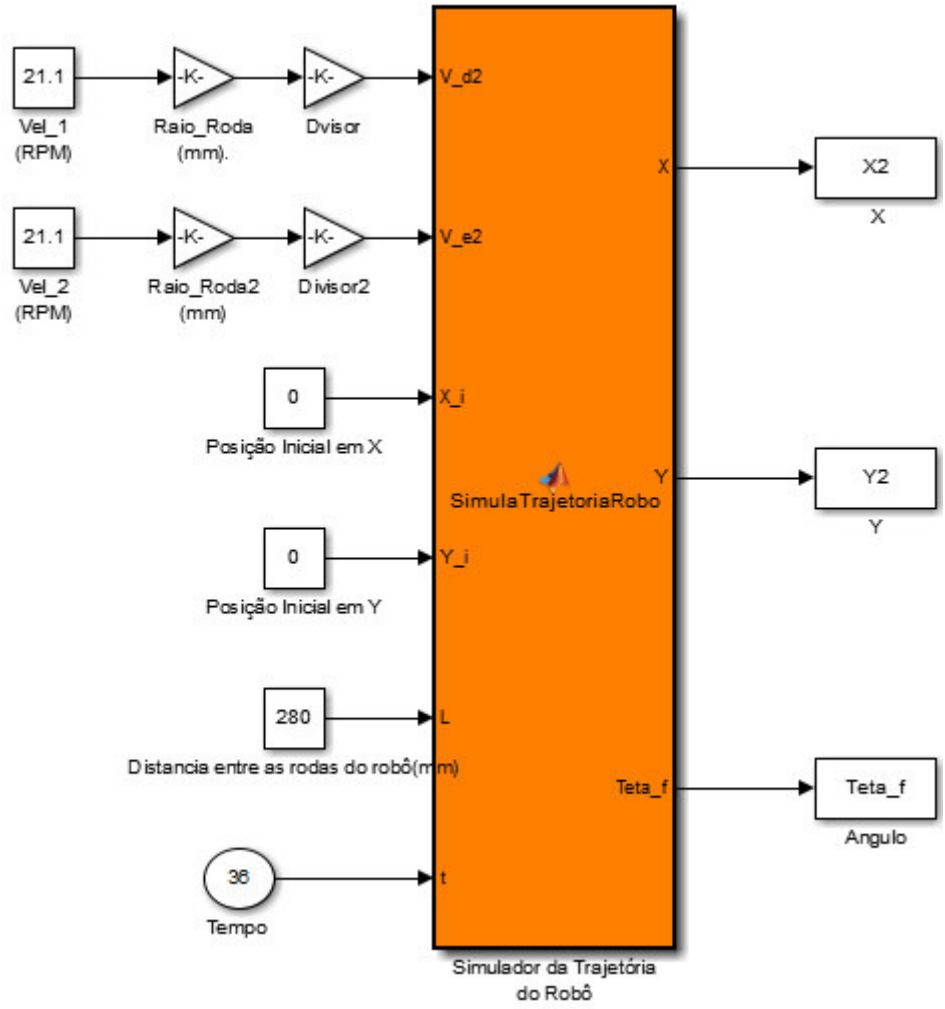


Figura 5.17: Bloco da Trajetória.

Este bloco usa o script apresentado no item 4.2 e a equação 4.1 para entrada das velocidades. Na Figura 5.17, V_d2 é o valor da velocidade angular aplicada ao motor direito e V_e2 é a velocidade aplicada ao motor esquerdo. X_i é posição inicial do robô para eixo X e Y_i par ao eixo Y, Teta_i é o ângulo incial do robô, L é a distância entre as 2 rodas medidas em milímetros, e t é o tempo.

Este programa deverá traçar o percurso percorrido pelo robô, uma vez que foram configuradas suas características, gerando os pontos X e Y no decorrer do tempo da simulação

para em seguida na prática verificar se o robô está obedecendo ao trajeto simulado. Os valores simulados desenvolvidos darão a resposta de X e Y em milímetros. Na sequência, segue a explicação de como serão desenvolvidas cada trajetória.

5.4 Implementação do Simulador Virtual – Com o feedback do encoder

Para a implementação do simulador com o feedback do encoder foi necessário construir um bloco no Simulink capaz de ler as interrupções do microcontrolador para que, junto com o encoder, fosse capaz de visualizar a quantidade de pulso emitidos em um intervalo de tempo. Na Figura 5.18 mostra-se a visão geral do sistema.

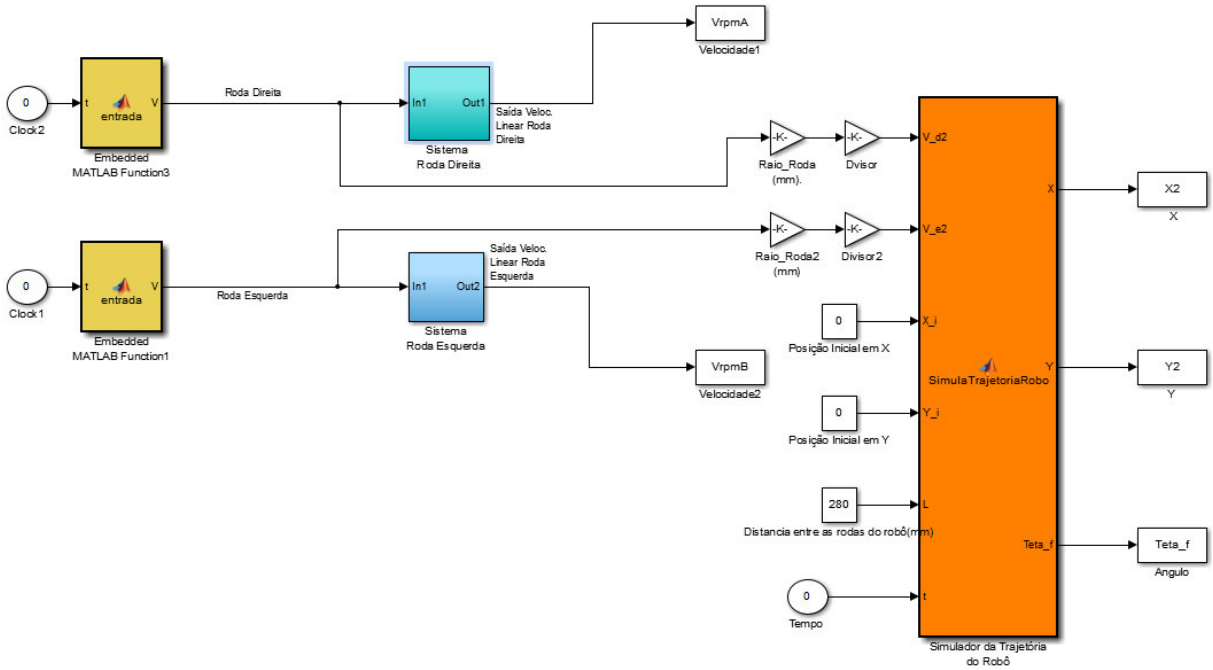


Figura 5.18: Simulador Virtual – MF.

A Figura 5.18 é semelhante ao sistema sem o feedback do encoder, já explicado na seção 5.3, sendo apenas diferente no bloco de controle do motor, onde é fechado o circuito com a leitura do encoder, e também no bloco de entrada, onde agora a entrada do sistema é feita através do valor desejado da velocidade diretamente em RPM. Estes dois blocos serão explicados a seguir.

5.4.1 Bloco de entrada

Na Figura 5.19 é apresentado o bloco de entrada da velocidade angular para o sistema.

Figura 5.19: Bloco de Entrada – MF.

Neste bloco a velocidade angular desejada é dada como referência para o sistema para que ele consiga chegar sozinho, através dos controladores, à velocidade desejada, tornando a saída do sistema igual a entrada.

5.4.2 Bloco de controle dos motores

A Figura 5.20 apresenta o bloco de controle de velocidade dos motores.

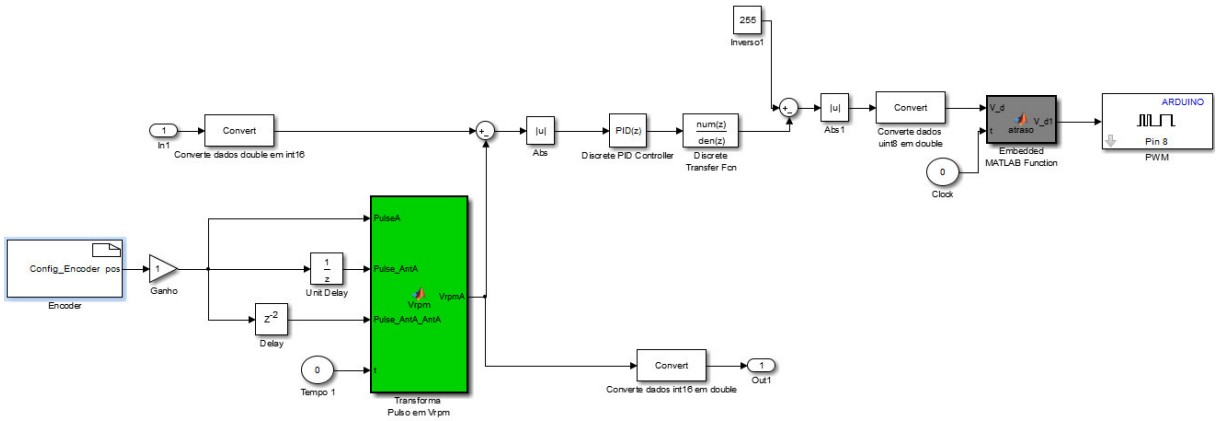


Figura 5.20: Bloco de controle dos motores – MF.

Para explicar melhor a Figura 5.20, foram realizados dois cortes na figura. O primeiro corte explica a leitura do encoder, e o segundo corte explica o controle do sistema.

Na Figura 5.21 mostra-se o bloco de aquisição de dados do sistema.

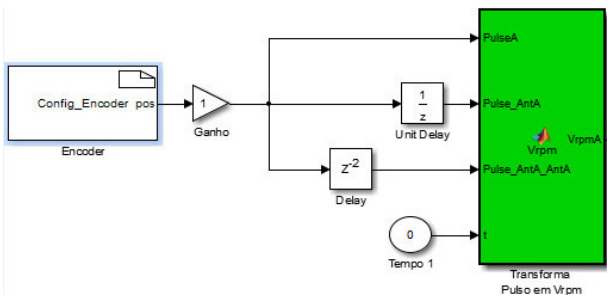


Figura 5.21: Bloco do Encoder.

No bloco do Encoder foi construído um sistema para contar a quantidade de pulso que é gerado pelo encoder sem intervalo de tempo. Para isto, o bloco encoder acessa duas portas de entrada de interrupções do microcontrolador, onde está ligado um encoder, sendo que existe um bloco deste para leitura da velocidade de cada motor.

Porém, considerando que o bloco Encoder conta somente a quantidade de pulsos e o objetivo é verificar a velocidade angular de cada motor, foi construído o bloco "V_{rpm}", que transforma a quantidade de pulsos emitidos pelo bloco anterior em velocidade angular. Esta transformação é realizada fazendo a média da quantidade de pulsos emitidos em um tempo presente em relação a um tempo anterior.

Para validação do modelo criado acima, realizaram-se comparações de diferentes velocidades medidas entre a Figura 5.21 e um tacômetro manual. Os dois apresentaram a mesma resposta, validando o sistema da Figura 5.21.

A Figura 5.22 apresenta o restante do controlador do sistema.

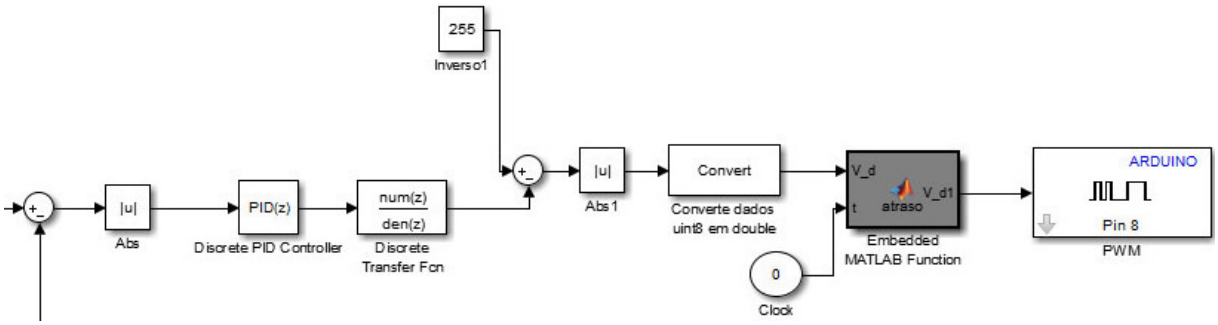


Figura 5.22: Controlador do Sistema em MF.

Na entrada deste sistema, verifica-se o erro fazendo-se a diferença entre valor da entrada (velocidade angular) com o valor medido pelo encoder. Se o erro não for nulo, o controlador irá ajustar a velocidade para que o valor da velocidade na saída seja o mais próximo possível da entrada

Como os valores em PWM no MatLab são invertidos em relação ao microcontrolador, na saída do sistema faz-se esta inversão, o que resulta na diferença de 255.

5.5 Implementação das trajetórias – Sem feedback do encoder

Utilizando o simulador já explicado na seção 5.3, testam-se as trajetórias do robô que serão válidas para todos os sistemas que precisarem programar um motor CC sem o uso de encoder, controlando-se de acordo com a metodologia que será vista a seguir.

5.5.1 Desenvolvimento das trajetórias

Para o desenvolvimento das trajetórias, será necessário conhecer antecipadamente os valores dos sinais de entradas e suas respectivas velocidades angulares, igual a Tabela 5.1.

Tabela 5.1: Velocidade de entrada conhecidas

Sinal de entrada	ω
198	21.1
150	37

Deste modo, utilizando a equação 4.1, sabe-se o valor do deslocamento de cada roda em mm a cada segundo (mm/s) que será a velocidade linear de cada roda.

5.5.2 Visualização das trajetórias reais desenvolvida pelo robô

Para visualizar as trajetórias percorridas pelo robô com base nas trajetórias simuladas na seção 4.3 é necessário efetuar a aquisição de dados do encoder. Para isto, foram necessárias modificações para programar direto na placa do microcontrolador, o que foi programado no MatLab, utilizando-se como parâmetro o sketch do arduino já desenvolvido por (Borges, 2014).

Desta forma é possível fazer a leitura do encoder a cada segundo, pelas portas de interrupções do microcontrolador. Assim, utilizou-se um módulo SD compatível com o Arduino para gravar os pulsos emitidos pelo encoder. Este programa pode ser visto no Apêndice G.

Com posse dos dados obtidos pelo encoder, é necessário transformar os valores de pulsos por segundo para a velocidade angular. Para isto utilizou-se a seguinte fórmula:

$$\omega = \frac{nº\ de\ pulsos}{96} * 60 * \frac{1}{84}$$

Onde nº de pulso é o número de pulsos obtido da leitura do encoder, 96 é a resolução do encoder utilizado, 60 é 60 segundos e 1/84 é devido ao uso do redutor alocado depois do encoder e antes do motor.

Finalmente para a visualização da trajetória real, converte-se a velocidade angular para velocidade milímetros por segundos, utilizando-se a equação 4.1 para, em seguida, efetuar a simulação conforme a script desenvolvido na seção 5.5.3.

5.5.3 Script para visualização da trajetória real percorrida pelo robô

Este Script foi desenvolvido no MatLab para ser possível visualizar o trajetória real que o robô percorre estando no chão ou suspenso no ar, a fim de identificar 2 situações.

A Primeira situação observar se a trajetória simulada é idêntica a trajetória real percorrida pelo robô no solo.

A segunda situação, é identificar a força de atrito entre o solo e a roda e escorregamento da rodas presentes no robô, comparando-se seu desenvolvimento no solo com seu desenvolvimento suspenso no ar.

A fim de possibilitar esta visualização, fez-se o armazenamento em uma planilha do excel, da quantidade de pulsos por segundos emitidos pelo encoder de cada uma das rodas do robô para, em seguida, encontrar a velocidade de cada roda conforme explicado na seção 5.5.2. Na Figura 5.23 é mostrado a imagem do script desenvolvido.

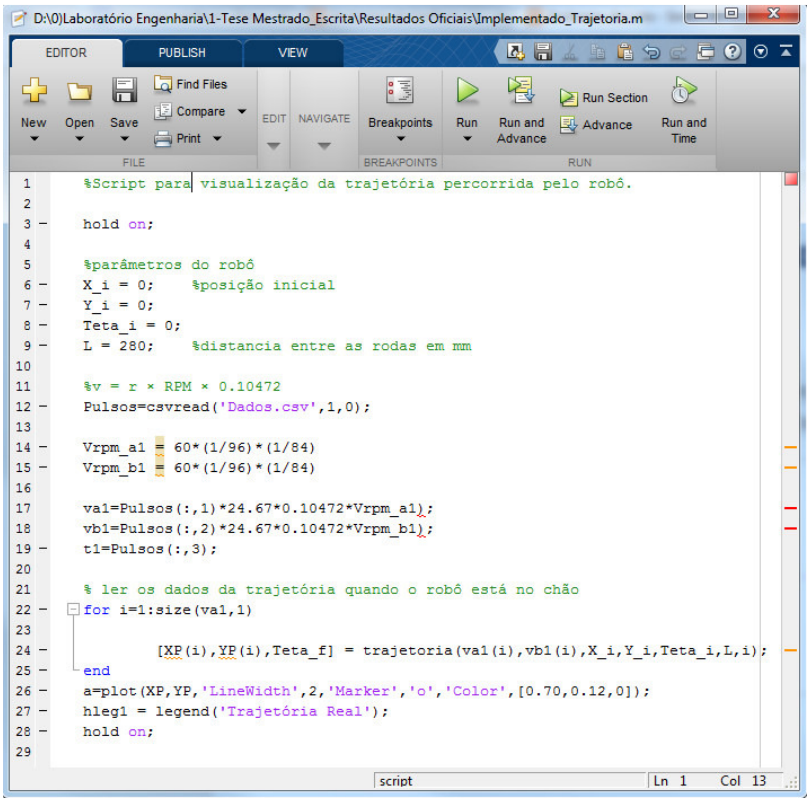


Figura 5.23: Script para visualização da trajetória real percorrida pelo robô.

A lista de código da Figura 5.23 pode-se apurar no Apêndice E.

5.5.4 Implementação da trajetória reta

- Sinal de Entrada = 198
- ω = 21.1

Para a primeira trajetória, será implementada a trajetória reta conforme simulação realizada na seção 4.3.1 para que o robô percorra durante um tempo de 19 s. Segue abaixo o resultado da velocidade de cada roda, mostrado na Tabela 5.2.

Tabela 5.2: Simulação trajetória real - reta

Roda	Sinal de Entrada	ω	$V_{mm/s}$
Direita	198	21.1	54,52
Esquerda	198	21.1	54,52

Conforme os valores da Tabela 5.2, tem-se como resultado a trajetória reta do robô, mostrado na Figura 5.24.

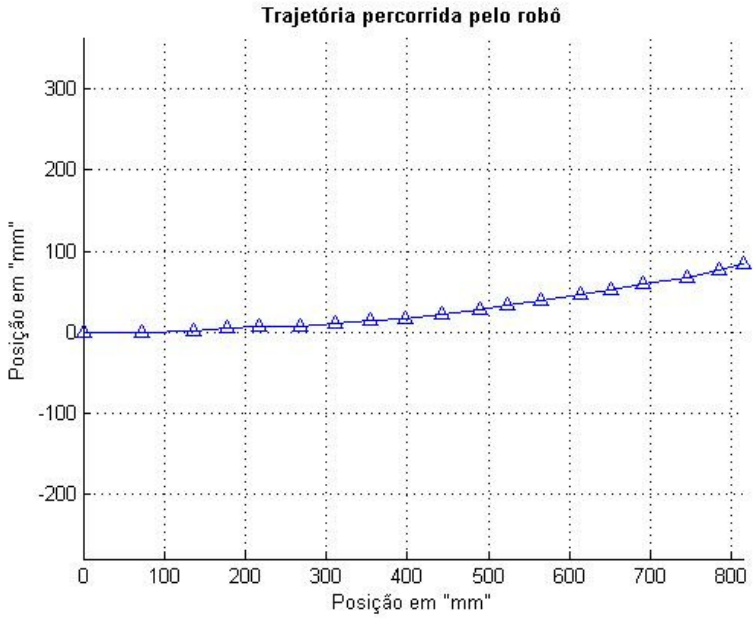


Figura 5.24: Trajetória real do robô – reta.

Nota-se uma leve inclinação na trajetória desenvolvida pelo robô, além do robô ter se deslocado uma distância de 800 milímetros dos 1000 milímetros propostos na simulação da seção 4.3.1.

A Figura 5.25 ilustra a imagem do teste realizado, na qual se pode constatar que a metodologia utilizada é válida. A Figura 5.25 ilustra o teste realizado com a distância de 1 metro.

Figura 5.25: Imagem do robô no teste da reta.

5.5.5 Implementação da trajetória circunferência

De acordo com a simulação realizada na seção 4.3.2, será implementada uma trajetória que irá percorrer um círculo com raio de 1m, em 34 segundos. Os valores utilizados podem ser vistos na Tabela 5.3.

Tabela 5.3: Simulação trajetória real - circunferência

Roda	Sinal de Entrada	ω	$V_{mm/s}$
Direita	16,4	61,8	159,68
Esquerda	68,1	81,4	210,33

A Figura 5.26 ilustra a trajetória real do robô para uma circunferência.

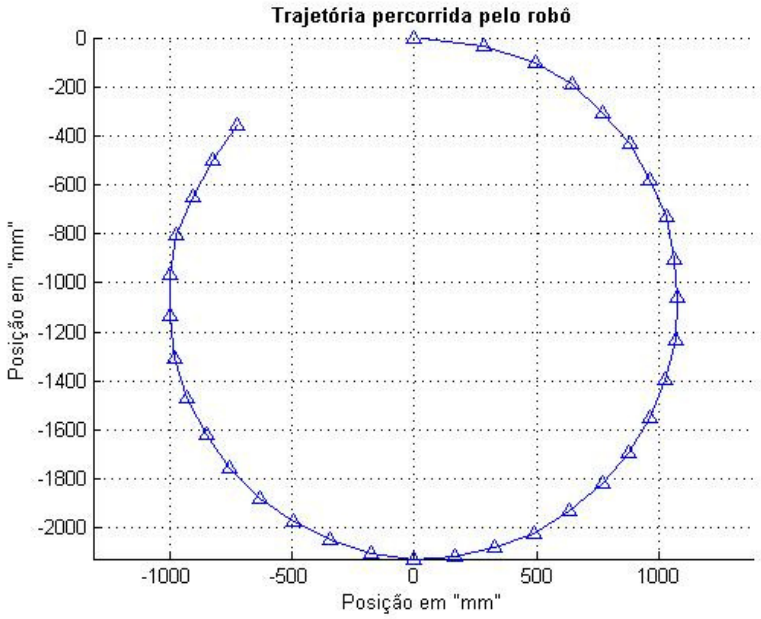


Figura 5.26: Trajetória real do robô – circunferência.

Na Figura 5.26 é possível verificar o correto deslocamento da circunferência com raio de aproximadamente 1 metro, porém com seu percurso não totalizado.

5.5.6 Implementação da Trajetória Circular

Nesta seção implementa-se a trajetória que servirá para o robô desviar de objetos, conforme já simulada na seção 4.3.3. Nesta situação, o robô deverá desviar do objeto com uma curva de profundidade 1 metro. Os valores utilizados podem ser vistos na Tabela 5.4.

Tabela 5.4: Simulação trajetória real - circular

Roda	Tempo (s)	Sinal de entrada	ω	$V_{mm/s}$
Direita	0-4	1	45	116,28
Esquerda	0-4	120	85	219,63
Direita	4-18	100	78	201,55
Esquerda	4-18	1	55	142,12
Direita	18-22	1	45	116,28
Esquerda	18-22	120	85	219,63

A Figura 5.26 ilustra a trajetória real do robô para efetuar uma curva.

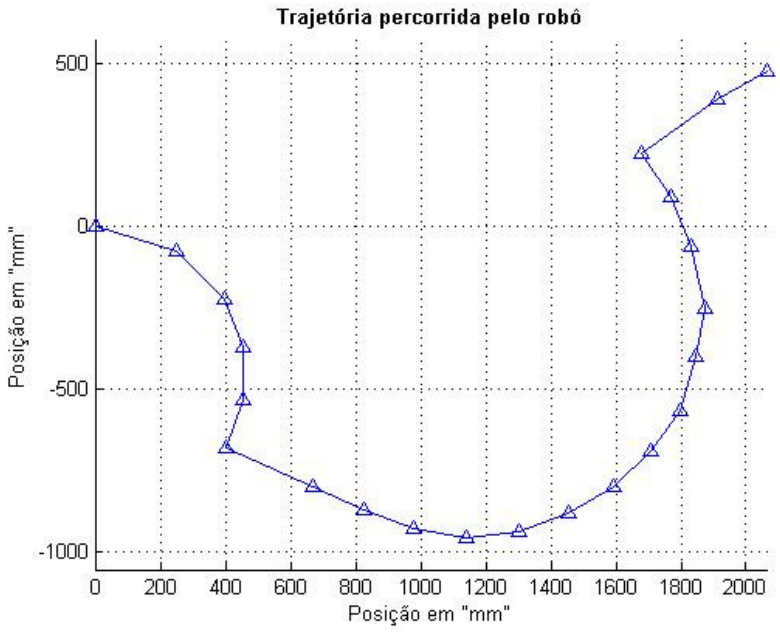


Figura 5.27: Trajetória real do robô – circular

No teste prático, observou-se que, com a programação para hipotético desvio de objetos, o robô seguiu seu caminho saindo do ponto inicial e voltou no mesmo sentido da trajetória inicial após efetuar uma curva de raio aproximadamente 1 metro de profundidade.

5.6 Implementação das trajetórias – Com feedback do encoder

De acordo com o simulador desenvolvido na seção 5.4, serão implementadas as trajetórias desenvolvidas na seção 4.3, tendo como o sinal de entrada, a própria velocidade angular utilizada para gerar as trajetórias.

A partir do percurso que o robô realizar da trajetória que foi simulada no MatLab, utiliza-se um script da seção 5.5.3 para visualizar os dados que serão armazenados em um bloco do Workspace, a fim de que colha os dados dessa trajetória e trace no computador em forma gráfica e se possa analisar se o percurso foi cumprido corretamente.

Vale dizer que os dados acima referidos foram obtidos de cada motor através do encoder, considerando os testes realizados com o robô suspenso (robô apoiado com as rodas livres).

5.6.1 Implementação da trajetória reta

Esta seção assemelha-se com aquela vista no item 5.5.4, com a diferença de que o sinal de entrada agora é dado pela própria velocidade angular desejada.

Tabela 5.5: Simulação trajetória real em MF - reta

Roda	Sinal de Entrada (ω)
Direita	21.1
Esquerda	21.1

Segue na Figura 5.28 o resultado da trajetória percorrida pelo robô.

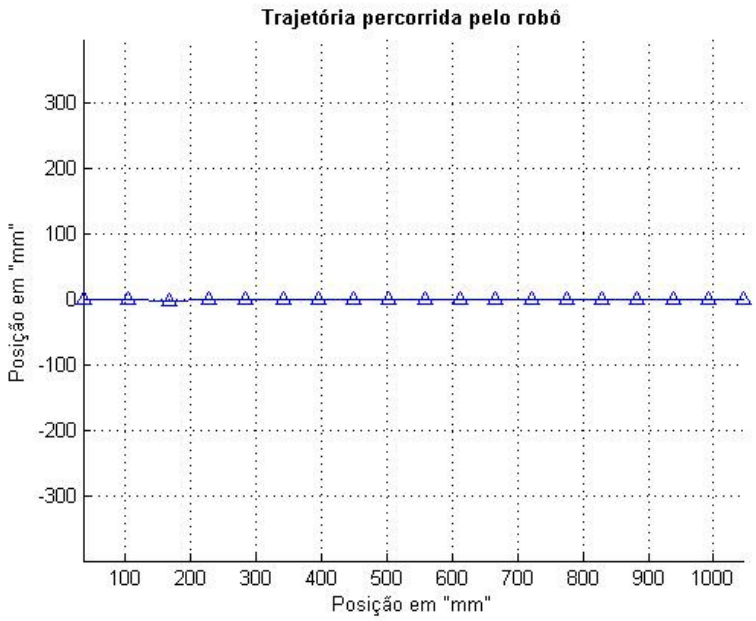


Figura 5.28: Trajetória real do robô em MF – reta.

Nota-se que com o sistema utilizando o feedback do encoder, os controladores responderam bem, percorrendo os 1000 mm em linha reta.

5.6.2 Implementação da trajetória circunferência

De acordo com a seção 5.5.5 para o sistema com feedback do encoder, será modificado apenas o sinal de entrada conforme a Tabela 5.6.

Tabela 5.6: Simulação trajetória real em MF - circunferência

Roda	Sinal de Entrada *(ω)*
Direita	81.4
Esquerda	61.8

A Figura 5.29 apresenta a trajetória real do robô para uma circunferência de raio 1 metro.

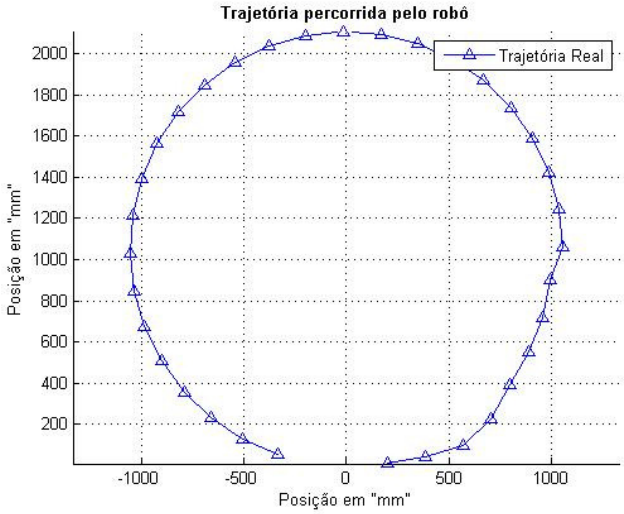


Figura 5.29: Trajetória real do robô em MF – circunferência.

Para a trajetória da circunferência, embora tenha percorrido a circunferência praticamente com raio de 1 metro, nota-se uma certa dificuldade nos controladores em estabilizar a velocidade da entrada na saída, mesmo sendo muito próximo a velocidade desejada, uma pequena alteração sobre a velocidade de cada roda faz surgir diferença na trajetória percorrida.

5.6.3 Implementação da Trajetória Circular

Nesta seção é tido como referência a seção 5.5.6, alterando-se somente o valor da entrada do sinal, em concordância com a Tabela 5.7.

Tabela 5.7: Simulação trajetória real MF - circular

Roda	Tempo(s)	Sinal de Entrada (ω)
Direita	0-4	85
Esquerda	0-4	45
Direita	4-18	55
Esquerda	4-18	78
Direita	18-22	85

Esquerda	18-22	45

A Figura 5.30 mostra-se a trajetória real do robô para efetuar uma curva.

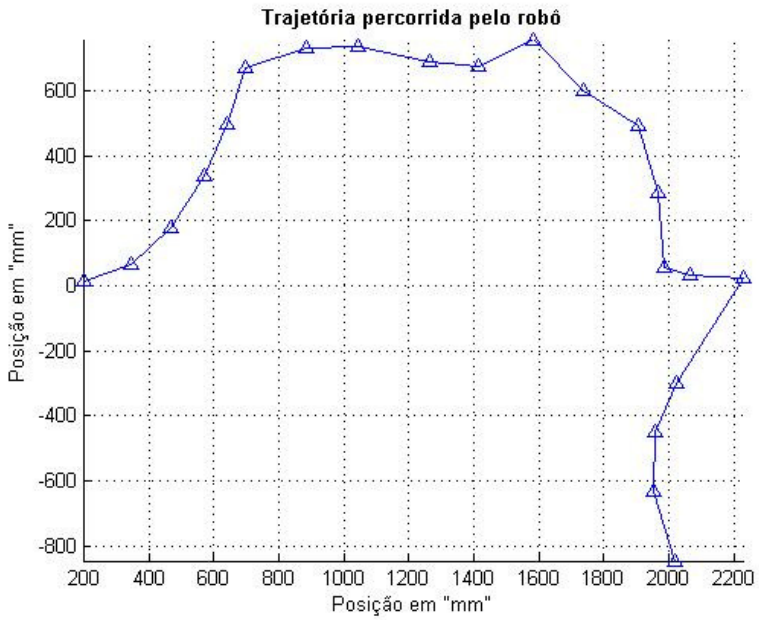


Figura 5.30: Trajetória real do robô em MF – circular.

Conforme a Figura 5.30, o robô desvia do objeto com raio menor que 1 metro e ultrapassa seu ponto de retorno em aproximadamente 600mm.

5.7 Considerações do Capítulo

Este capítulo apresentou a implementação da teoria desenvolvida neste trabalho, tanto para o modelo cinemático como dinâmico. Para o modelo dinâmico, testou-se a estabilidade do sistema sem e com o feedback do encoder. Já para o modelo cinemático, foram implementadas as trajetórias reta, circunferência e a trajetória circular para um sistema sem o feedback do encoder e para um sistema com o feedback do encoder.

No próximo capítulo será comparada a teoria (Simulação) com a prática (implementação) para verificar a proximidade do modelo desenvolvido.

6 COMPARAÇÕES DAS SIMULAÇÕES COM OS RESULTADOS EXPERIMENTAIS

Neste capítulo será analisado o erro de uma trajetória simulada com a trajetória real, comparando os valores simulados com os valores implementados.

Para os testes realizados sobre a estabilidade do sistema, serão comparados todos os valores da amplitude do sinal em relação ao seus correspondentes entre a simulação e a implementação. Por consequência, o erro será encontrado fazendo-se a subtração de cada ponto da amplitude do sistema simulado com o sistema implementado. O tempo de referência determinado para comparação entre a resposta simulada e implementada, é de 20 s.

Para ilustrar o erro da trajetória real em relação a trajetória simulada, será analisado o erro em todo o circuito percorrido pelo robô, fazendo-se a subtração de cada ponto da trajetória simulada com a implementada, utilizando-se em seguida para plotar o gráfico do sistema o comando do MatLab “errorbar”, que mostra os intervalos de confiança de dados ou o desvio ao longo de uma curva explicados na seção 6.2.1 com referência a Figura 6.3.

6.1 Script para comparação das trajetórias.

Este Script foi desenvolvido no MatLab para comparação da trajetória simulada, com a trajetória real percorrida pelo robô. Para possibilitar esta comparação, fez-se o armazenamento, da quantidade de pulsos por segundos emitidos pelo encoder de cada uma das rodas do robô numa planilha do excel. Esse pulsos foram obtidos do cartão SD conforme já explicado na seção 5.5.2.

Nessa linha de consideração, para comparar a trajetória simulada com a real, roda-se em primeiro momento a trajetória simulada, neste caso Gerador_trajetoria conforme script já mostrado na seção 5.3, para em seguida rodar no mesmo gráfico a trajetória real, conforme os valores de velocidades convertidos explicados na seção 5.5.2. Na Figura 6.1 visualiza-se o script desenvolvido.

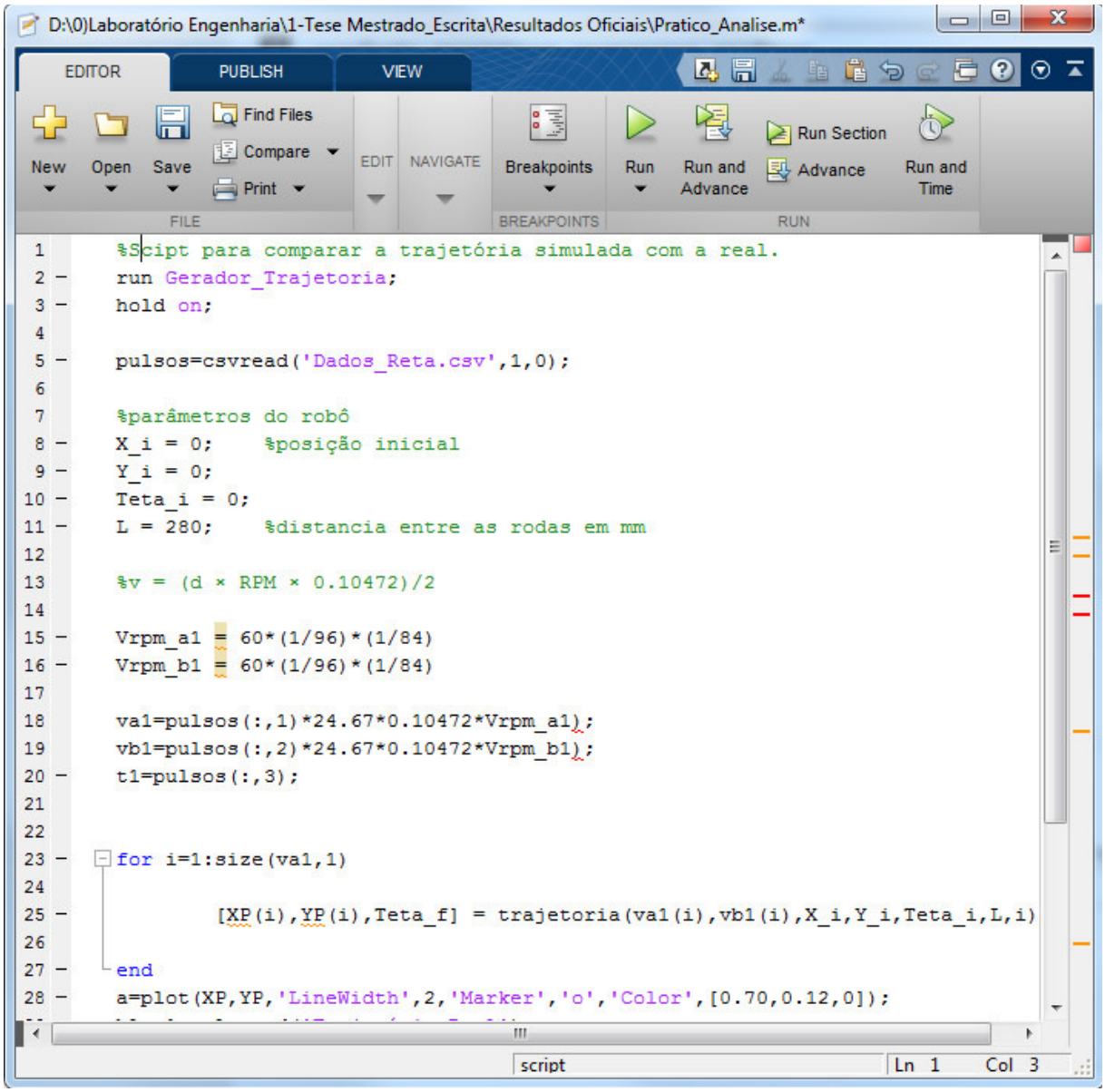


Figura 6.1: Script para comparar a trajetória real com a simulada.

O código do Script da Figura 6.1 pode-se averiguar no Apêndice F.

6.2 Comparação das Trajetórias – Sem feedback do encoder

Seguem abaixo as comparações das trajetórias entre o sistema simulado da seção 4.3 e o sistema implementado da seção 5.5 observando-se seu erro.

6.2.1 Trajetória Reta

Segue na Figura 6.2 a comparação entre a trajetória reta simulada e implementada.

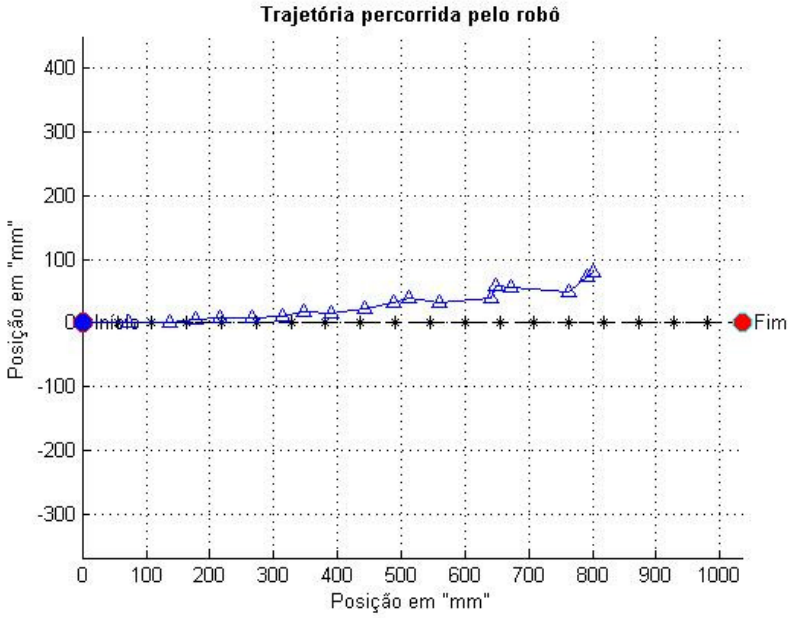


Figura 6.2: Trajetória reta – comparação.

Na Figura 6.2 são apresentadas duas trajetórias: a simulada representada pela linha tracejada e a trajetória real percorrida pelo robô, representada pela linha contínua. Desta forma é possível observar uma diferença entre as duas trajetórias, mormente porque a trajetória real percorre uma distância menor (cerca de 800 mm) da distância de 1000 mm simulada. Seu erro é apresentado na Figura 6.3.

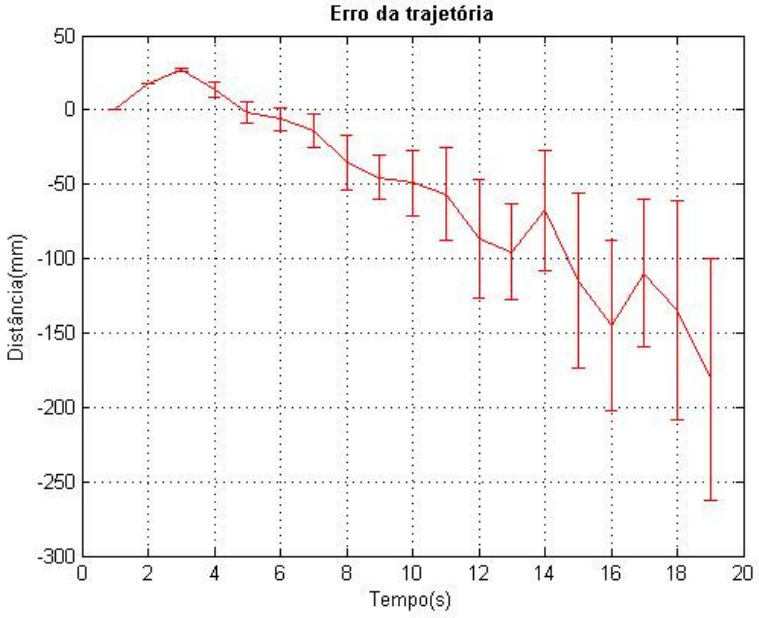


Figura 6.3: Erro da trajetória – reta.

De acordo com o gráfico do erro da Figura 6.3 verifica-se um erro crescente ao longo do tempo sendo que a linha horizontal representa o valor do erro em relação ao eixo “x” e a linha vertical em relação ao eixo “y, porém para o erro representado em “y” existem 2 valores, um acima da curva e um abaixo da curva, sendo este valor, o valor do erro de “y” para aquele ponto somado positivamente para o erro acima da curva e negativamente para o erro abaixo da curva com o valor do erro de “x”.

Neste caso, o valor do erro termina em -181 no eixo “x” significando que a trajetória real ficou com uma distância de 181mm atrasada em relação a trajetória real e no eixo “y” termina com um erro de 81.1mm, sendo este valor somado positivamente com -181 conforme já explicado acima, gera-se um valor no gráfico de -99.9 e somado negativamente com -181, gera-se o valor no gráfico no eixo y de -262.1mm. Assim, para o último ponto, conforme o gráfico do erro da Figura 6.3 apresenta um erro da trajetória real em relação a simulada de 181mm de diferença em relação ao eixo “x” e de 81.1mm de diferença em relação ao eixo “y”.

Esta mesma simulação foi realizada com o robô suspenso, verificando seu comportamento igual ao sistema simulado, conforme pode ser visto na Figura 6.4

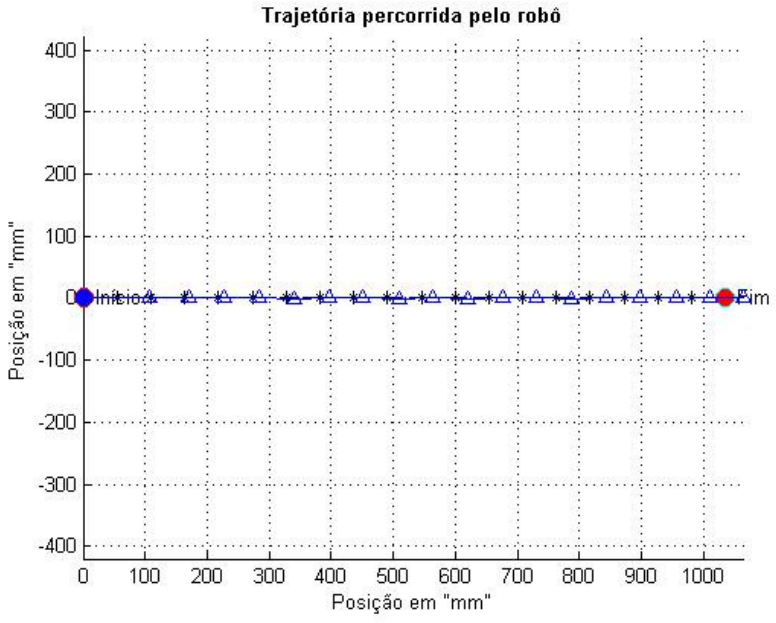


Figura 6.4: Trajetória reta – comparação com robô suspenso.

Desta forma é possível concluir que a diferença entre a trajetória real do robô com a trajetória simulada na Figura 5.24 é causada por conta das forças de atrito do solo com as rodas do robô e o escorregamento que ocorre nas rodas.

6.2.2 Trajetória Circunferência

Na Figura 6.5 demonstra-se a comparação da trajetória circunferência

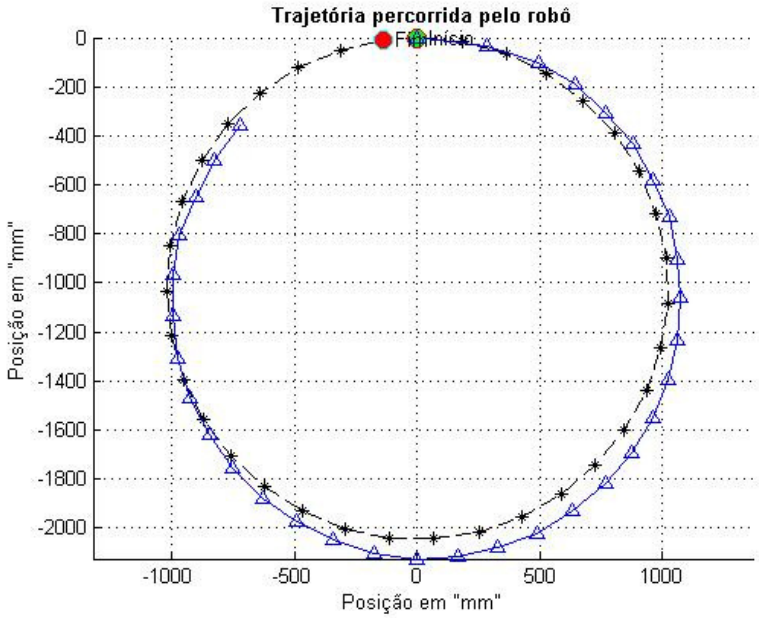


Figura 6.5: Trajetória circunferência – comparação.

Nesta trajetória nota-se uma similaridade entre a trajetória simulada e a real, porém a trajetória real acaba não completando totalmente o circuito definido na simulação. Observa-se na Figura 6.5 pontos inseridos ao longo da trajetória simulada e da trajetória real, onde cada ponto corresponde a 1 segundo. Seu erro é apresentado na Figura 6.6.

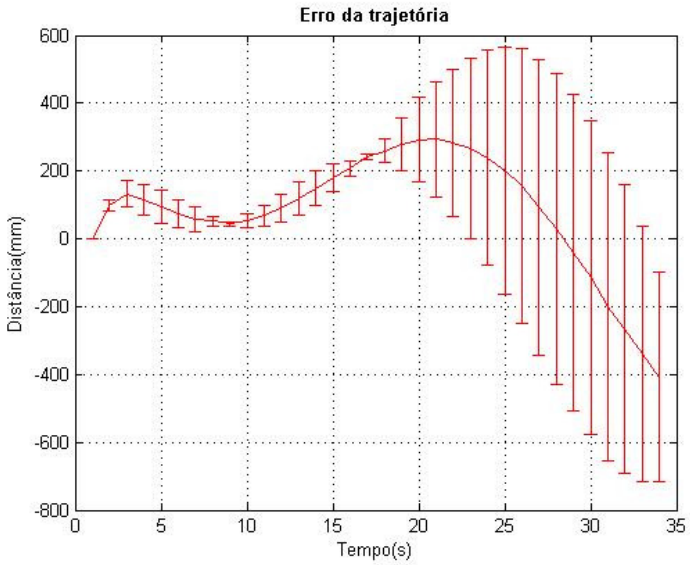


Figura 6.6: Erro da trajetória – circunferência.

No gráfico do erro da Figura 6.6 mostra-se um erro crescente até um determinado tempo e um erro decrescente em seguida significando alteração da velocidade ao longo do percurso. Para o eixo “x” houve uma variação máxima de 132 mm de erro entre a trajetória simulada e real. Para o eixo “y” houve uma variação máxima de 128 mm .

A trajetória real percorre a circunferência até o equivalente a 30 segundos da trajetória simulada, não completando o percurso por inteiro. Este resultado mostra um decaimento do torque do motor para velocidades mais altas.

6.2.3 Trajetória Circular

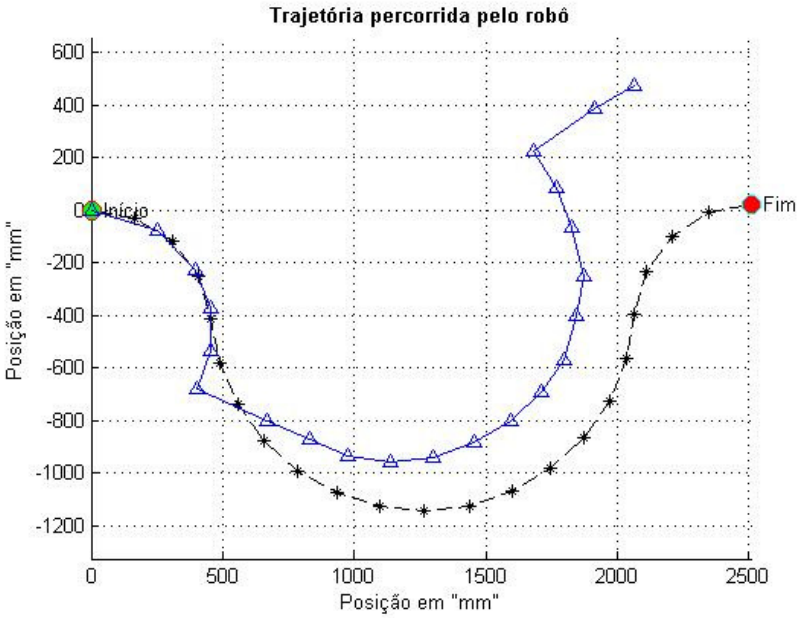


Figura 6.7: Trajetória circular – comparação.

Na Figura 6.7 observa-se uma grande diferença entre a trajetória simulada e a real, sendo a trajetória real percorre os mesmo pontos da trajetória simulada somente no inicio do percurso, ocasionando um erro para o restante do percurso, conforme pode ser visto na Figura 6.8.

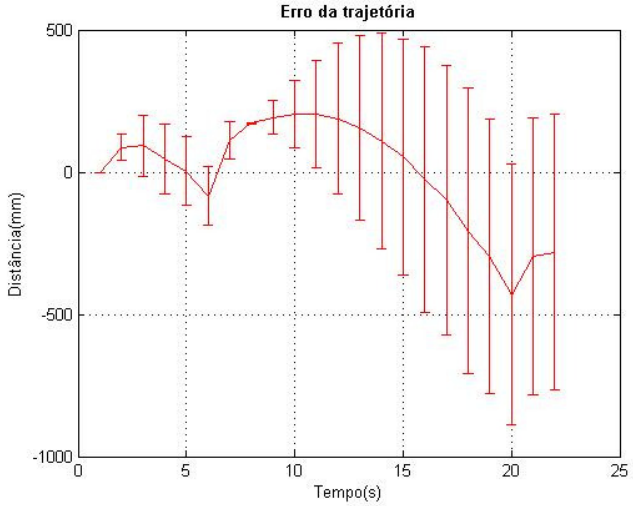


Figura 6.8: Erro da trajetória – circular.

Para o erro da Figura 6.8, nota-se um erro inicial baixo e posteriormente um erro alto, tanto em relação ao eixo “x” com para o eixo“y”. Para o erro em relação ao eixo “x”, após sexto segundo, a trajetória real se mantem adiantada em relação a trajetória simulada até o decimo primeiro segundo, decrescendo o erro até se tornar próximo de zero no decimo sétimo segundo, aumentando-se o erro posteriormente porém com agora a trajetória real atrasada em relação a trajetória simulada. Já para o erro em relação a trajetória “y” se torna cada vez maior após o décimo segundo, sendo claramente notado na Figura 6.7.

O erro na trajetória do robô em relação a trajetória simulada é causada pelas forças de atrito atuantes no sistema.

6.3 Comparação das Trajetórias – Com feedback do encoder

Nesta seção serão demonstradas as comparações das trajetórias entre o sistema simulado da seção 4.3, que se verá no gráfico em linha pontilhada, e o sistema implementado da seção 5.6, observado no gráfico em linha contínua.

Na sequência da comparação de cada trajetória será demonstrado o erro da trajetória.

6.3.1 Trajetória Reta

Na Figura 6.15 segue a comparação entre a trajetória reta simulada e implementada.

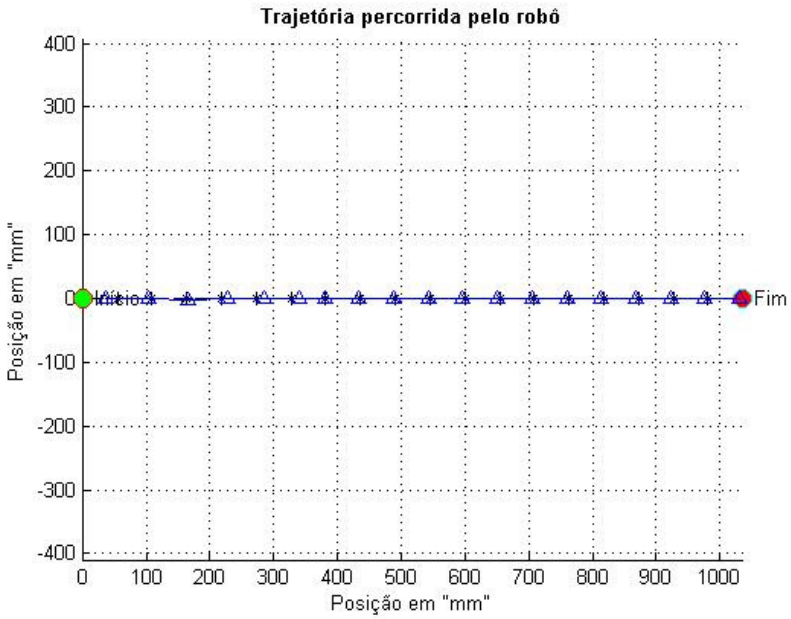


Figura 6.9: Trajetória reta em MF – comparação.

Neste sistema para a trajetória reta, o robô foi mais eficiente que o sistema sem o feedback do encoder, mostrado na Figura 6.2. Percebe-se que o robô percorreu a trajetória em linha reta, isto é, deslocou-se exatamente como o sistema simulado.

Na Figura 6.10 apresenta-se o erro da trajetória reta.

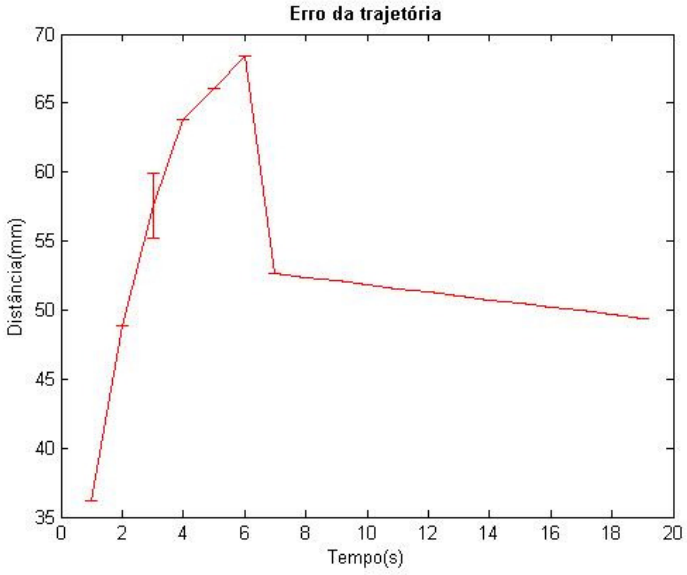


Figura 6.10: Erro da trajetória em MF – reta.

Na Figura 6.10 nota-se um erro muito pequeno para a trajetória reta, onde a maior diferença entre aquela simulada e a implementada está na diferença de tempo percorrida sobre cada ponto do sistema, sendo os pontos da trajetória implementada um pouco mais adiantados em relação a trajetória simulada, tendo um erro máximo de 66 mm em relação ao eixo “x” e para o eixo “y” o erro se manteve zero praticamente ao longo do tempo.

6.3.2 Trajetória Circunferência

Na Figura 6.11 mostra-se a comparação da trajetória circunferência

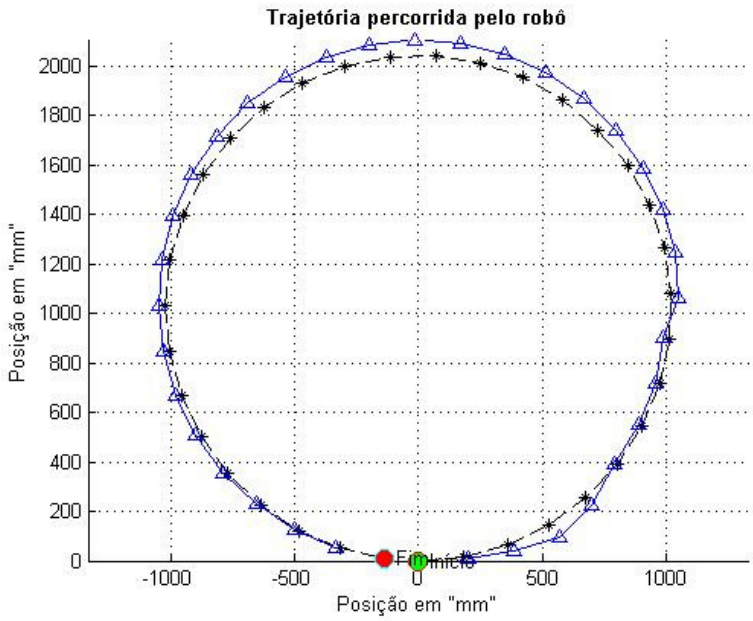


Figura 6.11: Trajetória circunferência em MF – comparação.

Para esta trajetória nota-se que em relação ao sistema sem o feedback do encoder, conforme a Figura 6.2, o robô percorreu toda a trajetória ficando muito próximo aos pontos do sistema simulado.

A diferença entre o sistema sem o feedback do encoder aqui apresentado e o sistema em com o feedback do encoder é que para aquele sistema realizou-se um estudo antecipado, manualmente e com o auxílio do tacômetro, dos valores de entrada do sistema para atingir as velocidades desejadas de cada motor. Em seguida, insere-se o sinal de entrada, que se relaciona com a velocidade angular desejada de casa trajetória.

Já para o sistema em com o feedback do encoder, simplesmente entra-se diretamente com a velocidade desejada, e o controlador do sistema terá que ajusta-la automaticamente, tentando ter na saída do sistema um sinal próximo igual ao da entrada. Por causa deste ajuste, para uma trajetória igual ao de uma circunferência, resulta-se em alguns desvios da trajetória simulada, porém o robô completou a circunferência com o raio de aproximadamente 1 metro.

Na Figura 6.12 ilustra-se o erro da trajetória circunferência.

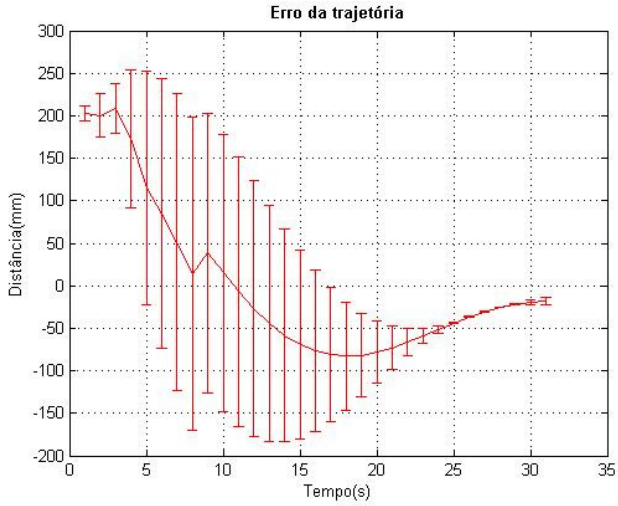


Figura 6.12: Erro da trajetória – circunferência.

O.erro apresentado na Figura 6.12 torna-se cresecente inicialmente por causa do controle PI para atingir a velocidade requerida pelo sistema, onde posteriormente o erro começa decrescer com o tempo.

6.3.3 Trajetória Circular

Na Figura 6.13 ilustra-se a comparação da trajetória circular.

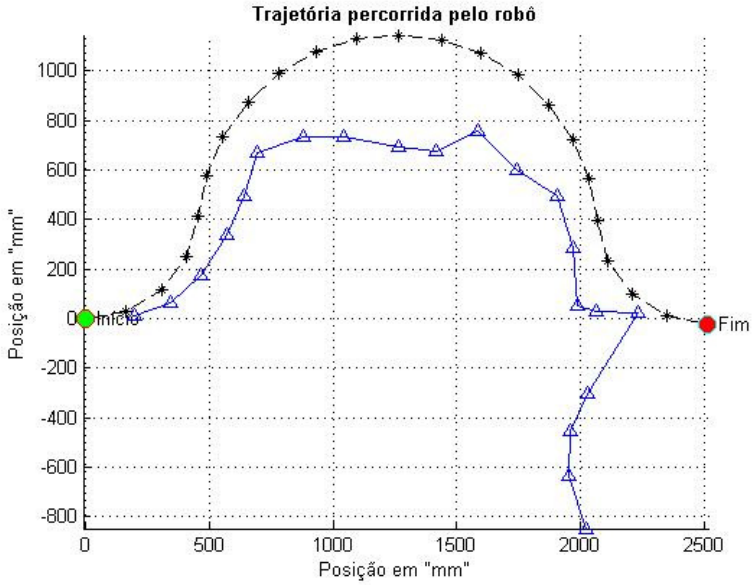


Figura 6.13: Trajetória circular – comparação.

A trajetória circular é a mais crítica do sistema, pois exige do sistema uma resposta rápida a mudança de velocidades. Como o controlador ajusta e estabiliza a velocidade do sistema para cada alteração, nesta trajetória um pequeno atraso sobre cada mudança de velocidade resultou em grande desvio da trajetória simulada.

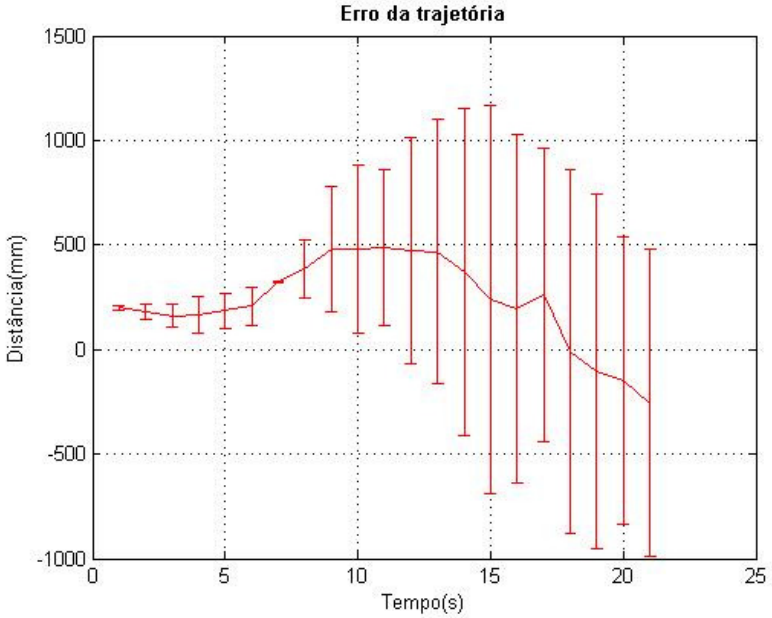


Figura 6.14: Erro da trajetória – circular.

Este erro é parecido com o erro apresentado para a trajetória desenvolvida no sistema em circuito aberto, já apresentada na seção 6.2.3.

Devido ao fato de o controlador não adequar rapidamente a mudança da velocidade com sua estabilização, resultam-se grandes erros em relação a trajetória simulada.

Para este sistema, somente o controlador não foi o suficiente. É necessário utilizar mais de um sistema de ajuste para conseguir percorrer na prática a mesma trajetória simulada.

6.4 Estabilidade do Sistema sem o Feedback do Encoder

Este foi o primeiro sistema analisado no capítulo 5, desta forma será verificado o erro entre a teoria e prática, mostrados na Figura 6.15: Comparação a Resposta ao Degrau – Velocidade.

6.4.1 Resposta ao Degrau – Velocidade

A Figura 6.15 ilustra a comparação da resposta ao Degrau entre simulada e praticada.

Tabela 6.1: Comparação – Reposta ao Degrau - Velocidade

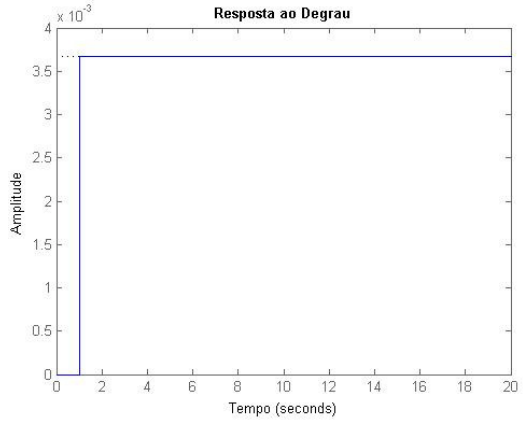
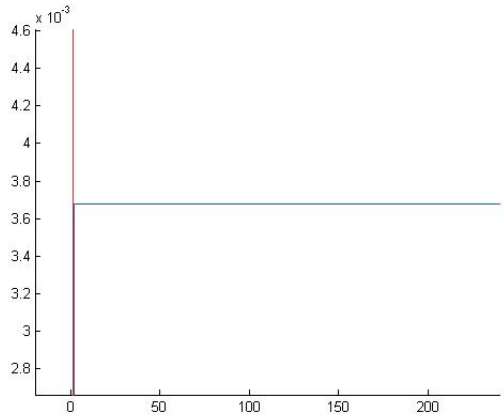
Teoria	Prática
	

Figura 6.15: Comparação a Resposta ao Degrau – Velocidade.

Salvando todos os valores da resposta em uma matriz, tem-se o seguinte sistema a ser analisado.

1. Simulação

A Figura 6.16 apresenta a resposta ao degrau para a velocidade.

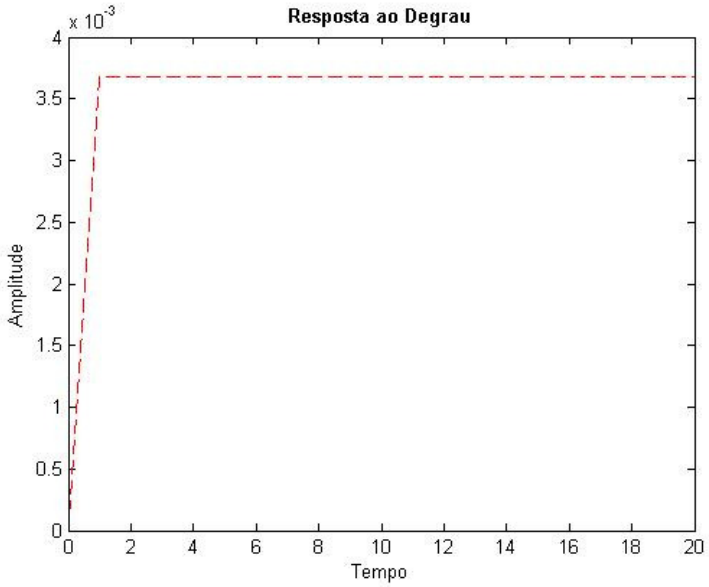


Figura 6.16: Simulação – Resposta ao Degrau – Velocidade.

Na Figura 6.16 cada ponto tracejado, significa um valor da resposta ao degrau que está salva em uma matriz para comparação com o sistema praticado.

2. Motor CC

A Figura 6.17 ilustra a resposta ao degrau para a velocidade.

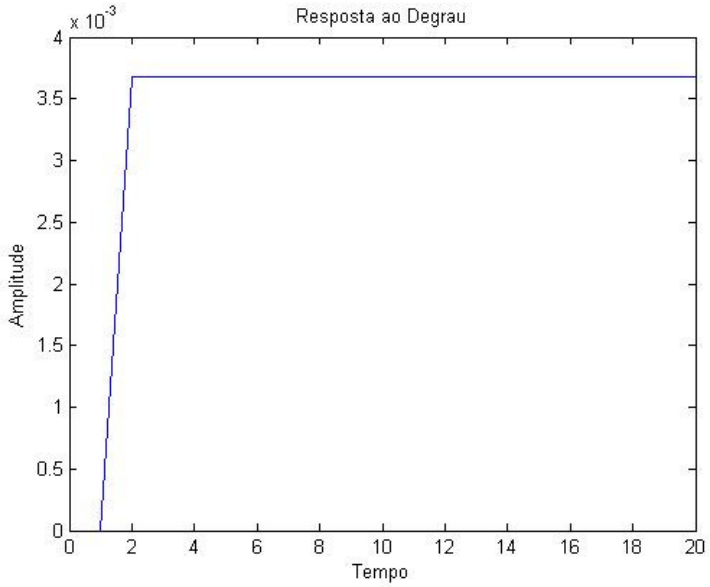


Figura 6.17: Implementado – Resposta ao Degrau – Velocidade.

A Figura 6.17 é a resposta que os motores CC. apresentaram ao sinal degrau.

Fazendo-se a subtração dos valores dos pontos do gráfico da Figura 6.16 com os valores da Figura 6.17, encontra-se o erro para cada ponto. Destarte, laçando-se um gráfico, observa-se o comportamento do erro, que pode ser vista na Figura 6.18.

3. ERRO

A Figura 6.18 ilustra o erro da resposta ao degrau.

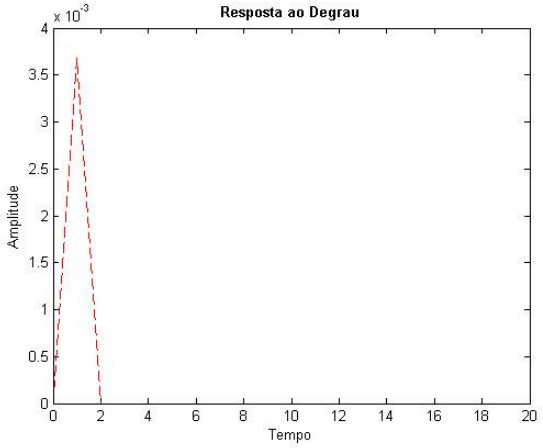


Figura 6.18: Erro da Resposta ao Degrau – Velocidade.

A Figura 6.18 ilustra o erro do sistema a resposta ao degrau para a velocidade.

Analisando-se a sequência acima, observa-se que a linha tracejada é o erro do sistema em resposta ao degrau. Nota-se que existe um erro somente nos primeiros dois segundos, porquanto há um atraso de resposta natural entre o microcontrolador e o computador (PC – Simulink) de 1s.

Este atraso faz com que o motor CC dispare assim que é ligado no primeiro 1 s, pois como existe este atraso faz a resposta ao motor nos primeiros 1 s ficar com o valor 0, porém os valores do PC em relação ao microcontrolador estão invertidos, o motor disparar na velocidade máxima no primeiro 1 s.

O tempo de trabalho do controle fica naturalmente estipulado a partir do 2º segundo, cujo valor coincide com a teoria, sem gerar o erro como pode ser visto na Figura 6.18.

6.4.2 Resposta ao Degrau – Angulo

A Figura 6.19 ilustra a comparação da resposta ao Degrau entre simulação e pratica realizado pelo sistema para o ângulo.

Teoria	**Prática**

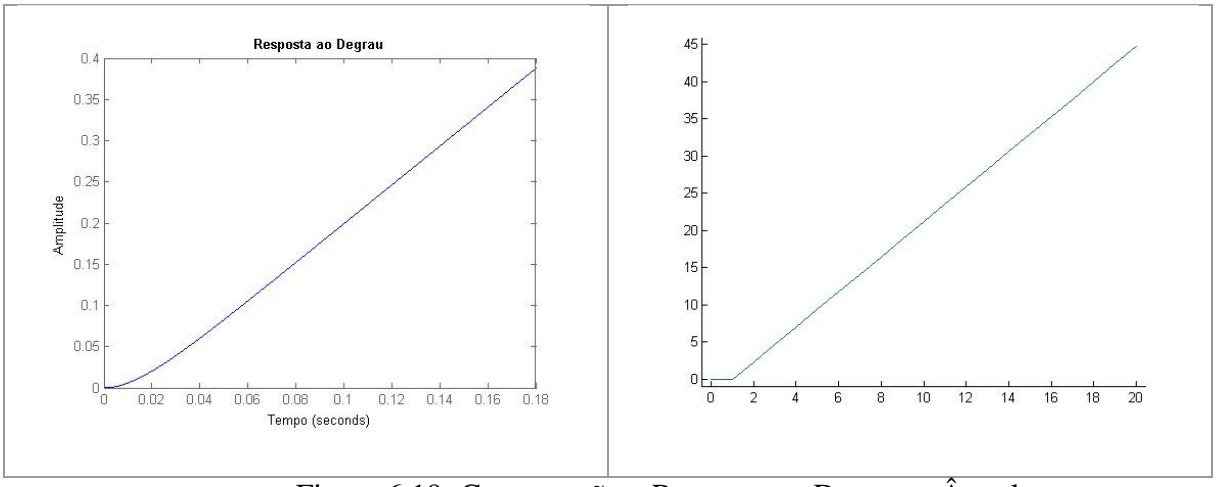


Figura 6.19: Comparação a Resposta ao Degrau – Ângulo.

De acordo com os passos já explicados na seção 6.4.1, tem-se:

1. Simulação

A Figura 6.20 ilustra a simulação da resposta do sistema ao degrau para o ângulo.

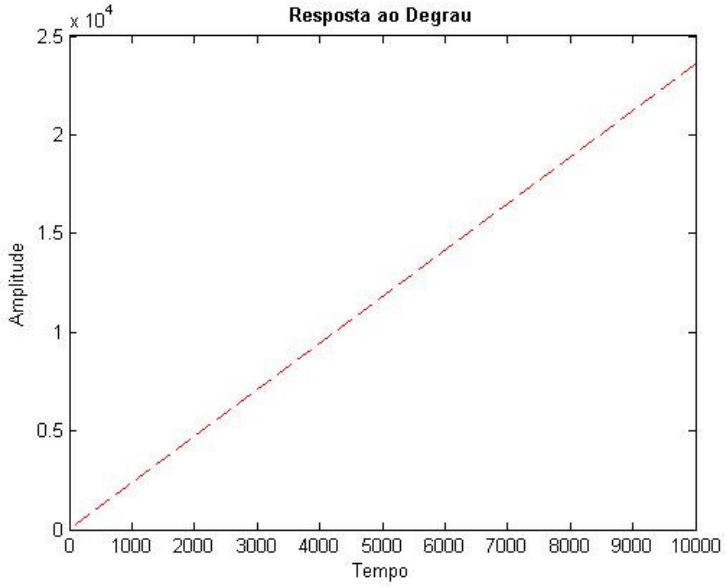


Figura 6.20: Simulação – Resposta ao Degrau – Ângulo.

2. Motor CC

A Figura 6.21 ilustra a implementação da resposta do sistema ao degrau para o ângulo.

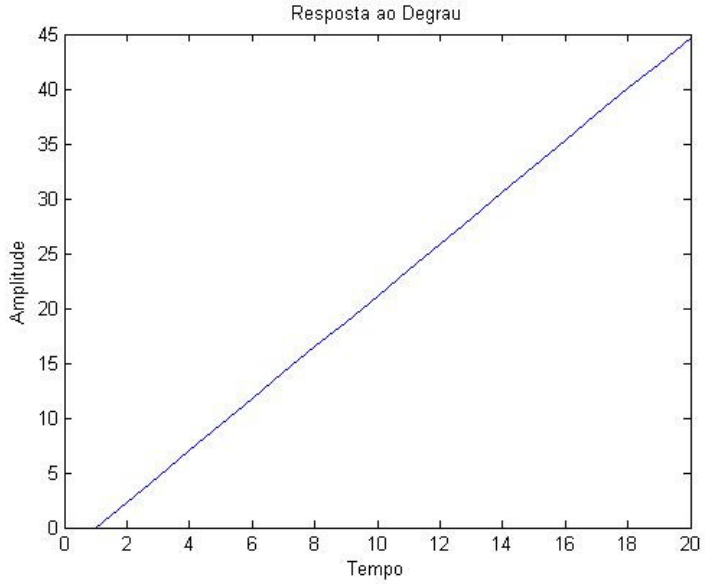


Figura 6.21: Implementado – Resposta ao Degrau – Ângulo.

3. ERRO

A Figura 6.22 ilustra erro do sistema a resposta ao degrau para o ângulo.

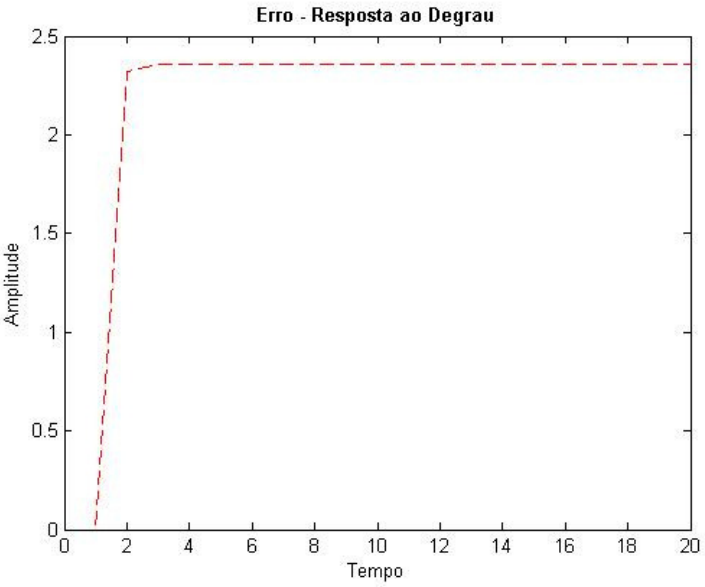


Figura 6.22: Erro - Resposta ao Degrau – Ângulo.

Como é possível avaliar, na Figura 6.22 houve um erro contínuo com uma amplitude menor que 2.5. Os dois sistemas simulados e implementados responderam iguais, mas devido ao atraso do sistema implementado, houve um erro contínuo desta diferença de tempo.

6.4.3 Resposta ao Impulso – Velocidade

A Figura 6.23 ilustra a comparação da reposta ao impulso do sistema para a velocidade.

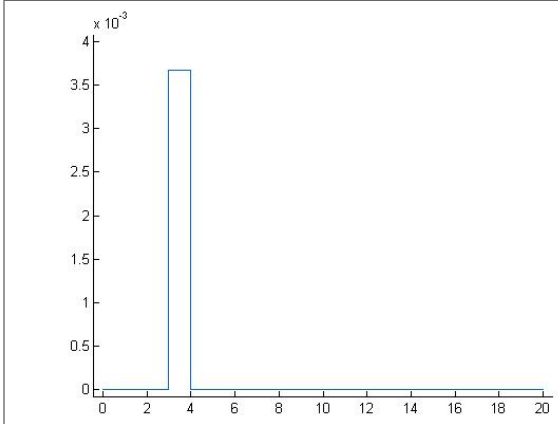
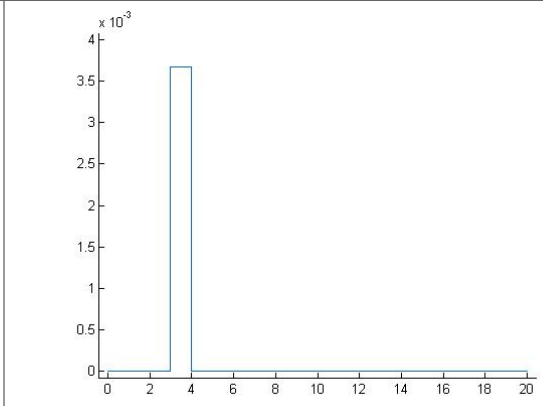
Teoria	Prática
	

Figura 6.23: Comparação a Resposta ao Impulso- Velocidade.

1. Simulação

O gráfico da simulação da resposta pode ser visto na Figura 6.24.

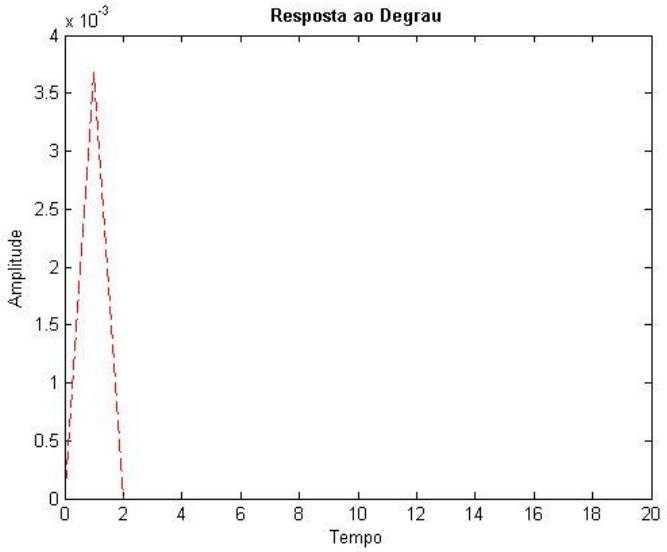


Figura 6.24: Simulação – Resposta ao Impulso – Velocidade.

2. Motor CC

A Figura 6.25 ilustra o gráfico da resposta ao impulso para a velocidade.

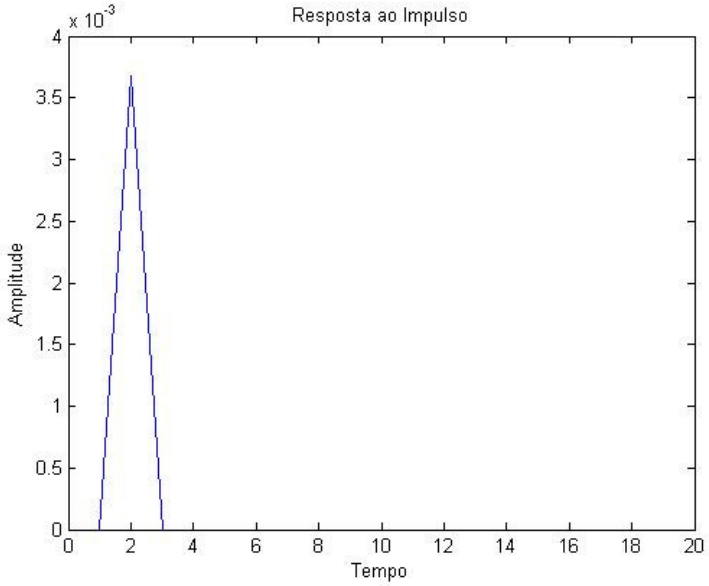


Figura 6.25: Implementado – Resposta ao Impulso – Velocidade.

3. ERRO

O gráfico do erro do sistema à resposta ao impulso pode ser visto na Figura 6.26

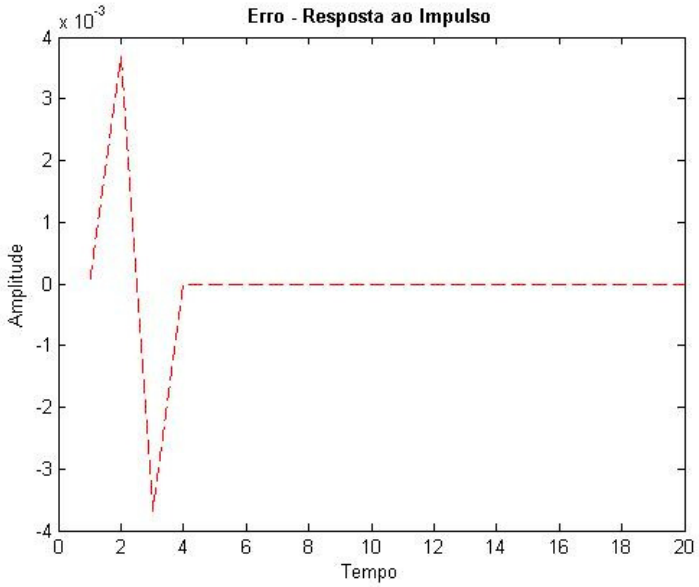


Figura 6.26: Erro – Resposta ao Impulso – Velocidade.

Percebe-se, então, que a Figura 6.26 mostra um erro gerado até o tempo 4 s e depois eliminado. Este erro ocorre pelo fato de o sistema ter fornecido a resposta após 2 s gerando o primeiro gráfico do erro positivo por causa do sistema que foi simulado ter aplicado o degrau

em 1 s e o implementado neste tempo ainda estava em zero, e depois que o simulado zero o impulso, o sistema implementado aplicou-o gerando o erro negativo, conforme a Figura 6.26.

6.4.4 Resposta ao Impulso – Angulo

A Figura 6.27 ilustra a comparação visual da reposta ao impulso do sistema para o ângulo

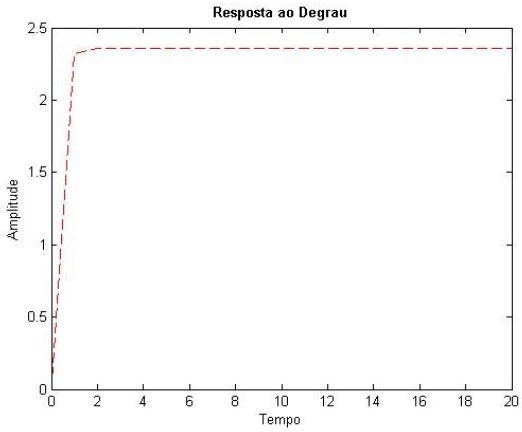
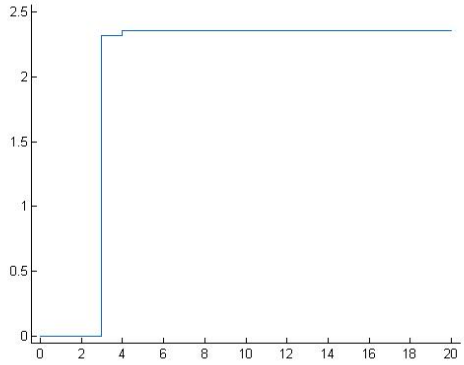
Teoria	Prática
	

Figura 6.27: Comparação a Resposta ao Impulso – Ângulo.

1. Simulação

A Figura 6.28 ilustra a simulação da resposta do sistema ao impulso para o angulo.

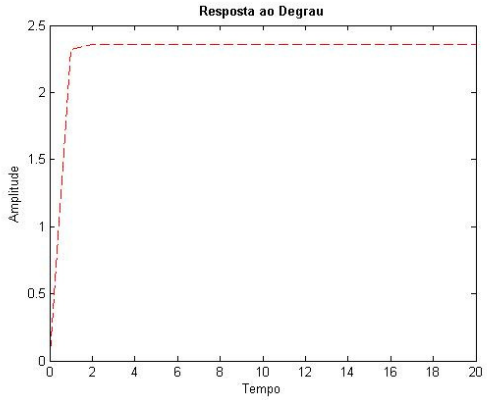


Figura 6.28: Simulação – Resposta ao Impulso – Ângulo.

2. Motor CC

A Figura 6.29 ilustra a implementação da resposta do sistema ao impulso para o ângulo.

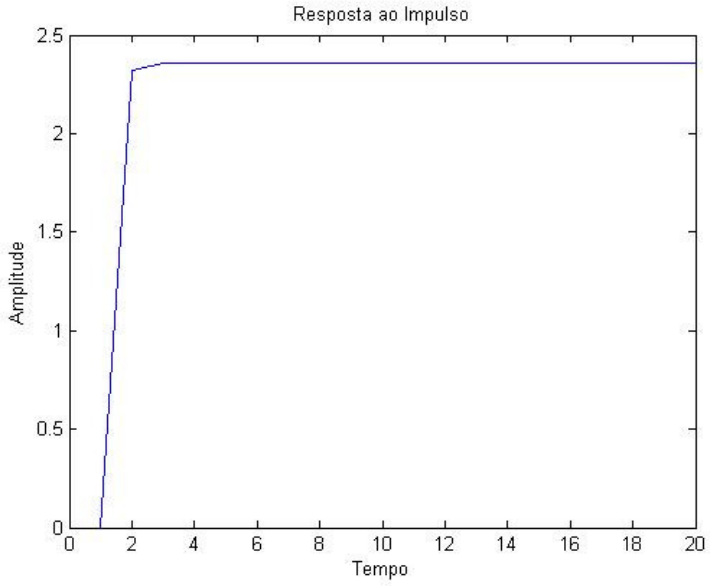


Figura 6.29: Implementado – Resposta ao Impulso – Ângulo.

3. ERRO

A Figura 6.30 ilustra o erro do sistema a resposta ao impulso para o ângulo.

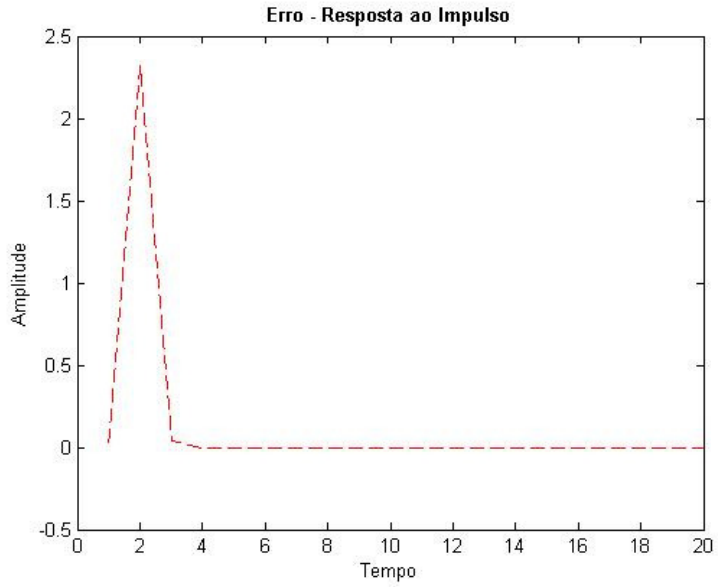


Figura 6.30: Erro – Resposta ao Impulso – Ângulo.

Conforme consta da Figura 6.30, houve um erro até 2 s, por causa do atraso a resposta no microcontrolador, mais que foi rapidamente extinguida.

6.5 Estabilidade do sistema Com Feedback do Encoder – Controle PI

Seguindo o desenvolvimento efetuado no item 6.1, será encontrado o erro para um sistema utilizando controlador proporcional integrador.

6.5.1 Resposta ao Degrau – Velocidade

Segue abaixo a resposta ao Degrau para velocidade.

A Figura 6.31 ilustra a comparação a reposta ao degrau do sistema para a velocidade.

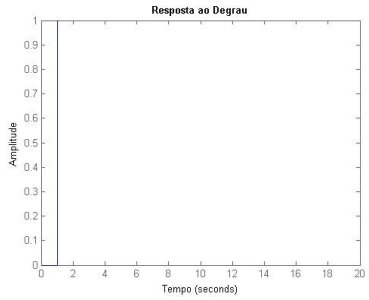
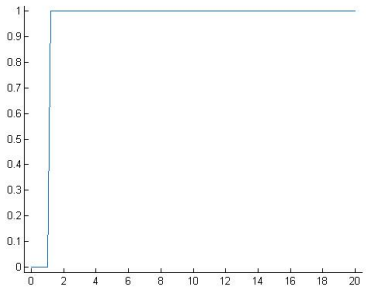
Teoria	Prática
	

Figura 6.31: PI-Comparação a Resposta ao Degrau – Velocidade.

1. Simulação

A Figura 6.32 ilustra a resposta ao degrau para a velocidade.

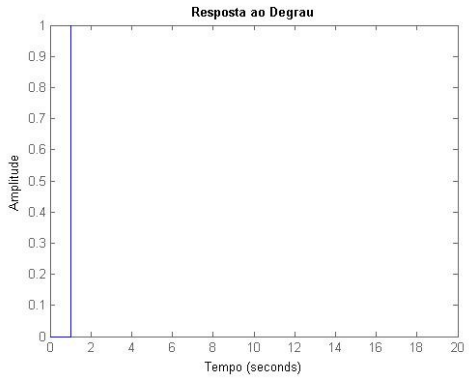


Figura 6.32: Simulado PI – Resposta ao Degrau – Velocidade.

A Figura 6.32 ilustra a simulação da resposta do sistema ao degrau para a velocidade com controlador PI.

A Figura 6.33 ilustra a implementação da resposta do sistema ao degrau para a velocidade.

Motor CC

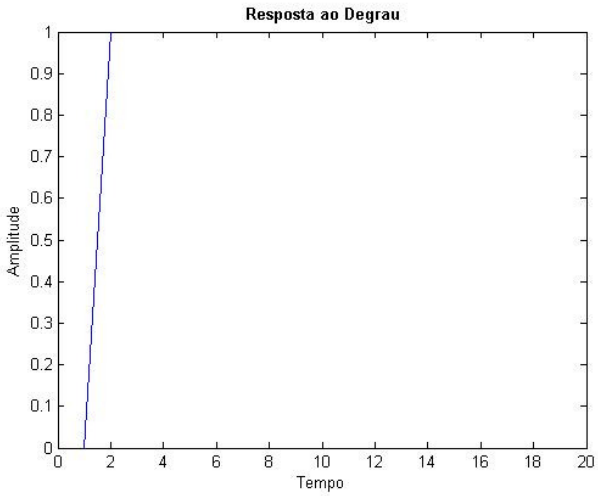


Figura 6.33: Implementado PI – Resposta ao Degrau – Velocidade.

2. ERRO

A Figura 6.34 Figura 6.34 ilustra o erro do sistema a resposta ao degrau com controlador PI para a velocidade.

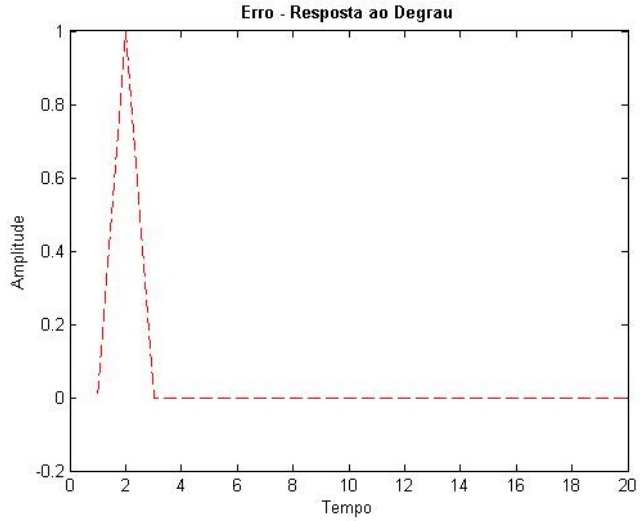


Figura 6.34: Erro – Resposta ao Degrau – Velocidade.

Da análise acima, percebe-se um erro inicial, e passados alguns segundos o erro no gráfico se torna zero. Este erro inicial está relacionado com um pequeno atraso no sistema implementado em relação ao sistema simulado.

6.5.2 Resposta ao Impulso– Velocidade

A Figura 6.35 ilustra a comparação a reposta ao impulso do sistema para o ângulo.

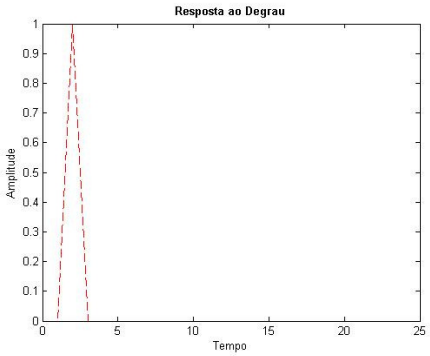
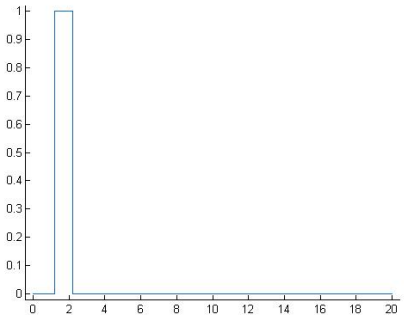
Teoria	Prática
	

Figura 6.35: PI-Comparação a Resposta ao Impulso – Velocidade.

1. Simulação

A Figura 6.36 ilustra a simulação da resposta do sistema ao impulso para a velocidade com controlador PI.

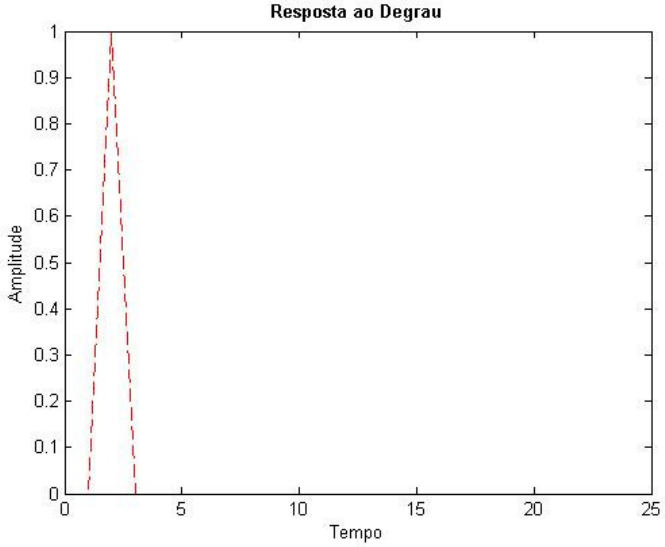


Figura 6.36: Simulado PI – Resposta ao Impulso – Velocidade.

2. Motor CC

A Figura 6.37 ilustra a implementação da resposta do sistema ao impulso com controlador PI para a velocidade.

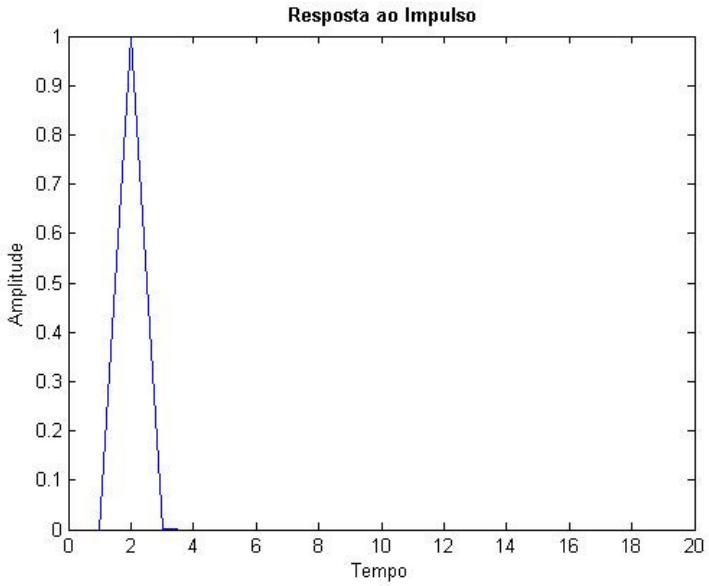


Figura 6.37: Implementado PI – Resposta ao Impulso – Velocidade.

3. ERRO

A Figura 6.38 ilustra o erro do sistema a resposta ao impulso com controlador PI para a velocidade.

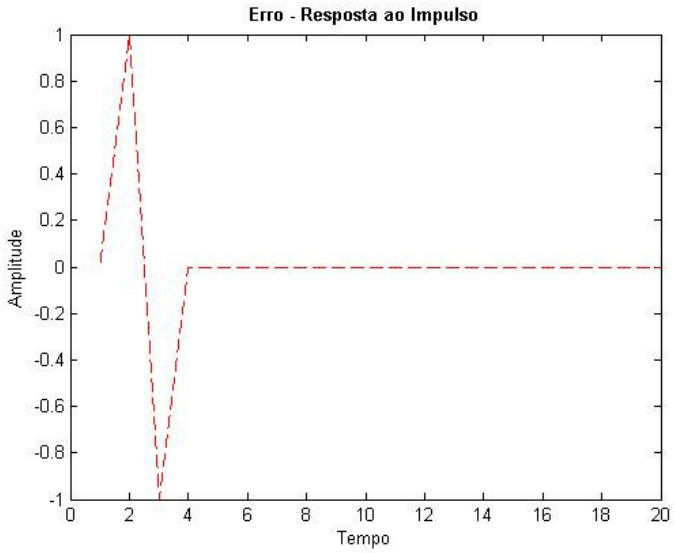


Figura 6.38: Erro – Resposta ao Impulso – Velocidade.

Para o impulso, houve um erro inicial por causa do pequeno atraso do sistema implementado em relação ao sistema simulado.

6.6 Considerações do Capítulo

Como se pode inferir nesta etapa, desenvolveu-se a comparação das trajetórias simuladas e implementadas para um sistema em sem e com feedback do encoder, relacionadas a cinemática.

Igualmente, em relação ao sistema dinâmico, foram apresentados gráficos das respostas ao degrau e impulso relacionadas a velocidade do motor CC para um sistema simulado e para um sistema implementado, fazendo-se comparações entre os dois sistemas através dos gráficos dos erros.

No capítulo seguinte serão apresentadas as conclusões finais do trabalho.

7 CONCLUSÕES

Neste trabalho criou-se um sistema controlável para o robô constituído de dois motores CC. Para isto fez-se uso da ferramenta MatLab para a criação de um simulador virtual que pudesse se comunicar em tempo real com o microcontrolador utilizado para controle dos motores CC.

Inicialmente, desenvolveu-se a teoria da cinemática direta conforme visto na seção 3.1 e simuladas três trajetórias na seção 4.1.

Também gerou-se a parte dinâmica na seção 3.4, onde se elaborou uma equação de estado completa. Nesta equação envolve-se além do equacionamento dos motores, a parte de ganho dos conversores do microcontrolador, do encoder e do redutor do motor, mostrando-se a parte final do equacionamento na seção 3.4.6

Uma vez encontrado o equacionamento do sistema, simularam-se as respostas do sistema ao degrau e ao impulso para o sistema sem feedback do encoder na seção 4.4, e com controlador PI na seção 4.6, enquanto que para com feedback do encoder e com controlador PI na seção 4.7.

Concomitantemente simularam-se várias configurações, como um sistema com ou sem redutor, com ou sem carga, tempo contínuo ou tempo discreto, com controlador e sem controlador, a partir das quais pode-se conferir que o sistema é estável para a *velocidade* para todas as simulações realizadas e instável para o *ângulo.* Estes resultados simulados, foram comprovados na prática, após implementar o sistema no robô e verificados os resultados que foram mostrados na seção 5.1 para o sistema sem feedback do encoder e na seção 5.2, o sistema com feedback do encoder e com controlador PI.

Após conseguir alcançar a estabilidade do sistema, foram encontrados os controladores proporcional e integrador para criar um sistema que chega-se o mais próximo possível do sinal de entrada, sendo esta parte mostrada na seção 4.7 para simulação e na seção 5.2 para a parte prática, averiguando-se que para os 2 processos houve a mesma resposta, o que comprovou o desenvolvimento.

Durante todo o trabalho, corroborou-se a teoria desenvolvida na prática, em seguida implementou-se além do controle dos motores CC, um sistema capaz de controlar a trajetória

do robô na prática. Assim sendo, com o uso de script e de blocos de controle, implementou-se o simulador virtual na seção 5.5 para controle do sistema sem feedback do encoder e na seção 5.6 implementou-se o simulador virtual para controle do sistema em com feedback do encoder, explicando-se a função de todos os blocos utilizados.

Realizadas as simulações junto ao sistema MatLab no capítulo 4, foram posteriormente implementas na prática como se pode conferir no capítulo 5, ocasião em que registraram-se as comparações entre os valores encontrados simulados e os valores encontrados implementados.

O capítulo 6, por sua vez, faz a comparação entre a parte simulada e a implementada para, em seguida, encontrar o erro por meio dos respectivos gráficos. Elaborou-se na seção 6.1 um script para fazer a comparação entre os dois sistemas e encontrar o erro da parte cinemática. Para o sistema sem feedback do encoder pode-se observar uma pequena discrepância – para algumas trajetórias -entre o modelo simulado e o real em virtude do atrito presente no solo com as rodas do robô, do escorregamento das rodas do robô e do torque do atuador. Para o sistema com feedback do encoder, observou-se que os controladores responderam bem para um sistema de baixo torque porém para um sistema de alta velocidades e com bruscas variações em intervalos pequenos de tempo, os controladores demoram mais que o tempo requisitado pela trajetória para se estabilizar, de acordo com o monitoramento realizado através do Simulink junto ao microcontrolador do sistema in-the-Loop.

Posto isso, na seção 6.4, orientada pela parte dinâmica, discutiu-se o erro da estabilidade encontrado para sistemas sem feedback do encoder relacionadas as resposta ao degrau e impulso sobre a velocidade e o ângulo. Já na seção 6.5 foi discutido o erro da estabilidade para o sistema com feedback do encoder e com controle PI relacionadas as resposta ao degrau e impulso somente para a velocidade que foi o sistema de referência utilizado na implementação. Desta forma obteve-se um erro gráfico nulo ou muito próximo de zero para todas comparações feitas entre os sistema simulados e implementados, concluindo-se pela correta modelagem do sistema.

Durante toda a execução do trabalho foram desenvolvidas pertinentes contribuições para o controle dinâmico e cinemático do robô. Em especial, pode-se mencionar o desenvolvimento de método de controle pela equação de estado, a partir da qual verifica-se o comportamento do sistema através da matriz de estado.

Em virtude de sua adaptabilidade, referido método de controle mostrou-se hábil para implementação em qualquer outro sistema, desde que sejam conhecidos os dados requeridos pela equação desenvolvida no Capítulo 3. Significa dizer, como já dito ao longo do trabalho, que se deve conhecer, seja por descrição do próprio fabricante ou descoberto por meios de softwares, as características do atuador (motor), do respectivo redutor se houver, caso contrário equivale a zero, do encoder se existir, senão equivale a zero e do microcontrolador (ou do ganho do conversor ADC)

Finalmente, conjugando o método de controle dinâmico do robô com o controle cinemático averiguado também ao longo do Capítulo 4, torna-se possível controlar a trajetória em qualquer outro sistema que utilize robô uniciclo, desde que modifique-se o valor de "L" (espaçamento entre as rodas do centro de cada roda) e "d" (diâmetro da roda) no script do MatLab visto neste trabalho.

7.1 Sugestões para Trabalhos Futuros

A partir do tema proposto nestes estudos, podem-se iniciar inúmeras pesquisas futuras..

Neste trabalho estabeleceu-se uma plataforma de baixo custo, o que favorece o acesso à pesquisa com implementação prática da teoria que se desenvolver.

O controle dinâmico foi abordado de uma forma objetiva, onde conforme a equação de estado desenvolvida, é possível adaptar diferentes pesos inseridos sobre o robô, além de atuadores, encoder e microcontrolador, pontos estes que podem ter exploração produtiva para a observação do comportamento do sistema em diferentes situações.

Almeja-se que, nos tempos que estão porvir, o desenvolvimento do modelo dinâmico deste trabalho tenha uma contribuição na solução de problemas relacionados ao controle de sistemas embarcados. No mesmo sentido, para sistemas que utilizem robôs uniclo, espera-que, além do controle dinâmico do sistema, se faça o controle da cinemática de acordo com a adaptabilidade dos scripts do MatLab junto com o Simulink desenvolvidos neste trabalho.

Portanto como recomendações para novos trabalhos de pesquisa, sugere-se investir no desenvolvimento e aperfeiçoamento do trabalho praticado pelo robô em questão, investigando seu modelo dinâmico e cinemático para situações diversas, como aplicação de peso em um

determinado momento da trajetória culminando ou não em algum desvio de trajetória, também questões ligadas ao atrito de tração.

Além disso, oportuno seria o desenvolvimento da comunicação sem fio entre o microcontrolador e computador (PC), através de módulos aplicáveis a placa Arduino, ou mesmo utilizando outros sistemas em conjunto, como o Raspberry.

Por fim cumpre mencionar que, com o futuro avanço no desenvolvimento do sistema, sugere-se a elaboração de um sistema em que, a partir da implantação de uma câmera no robô, seja possível reconhecer obstáculos em um ambiente, de modo que a trajetória programada oriente o robô a percorrer o trajeto identificando os obstáculos e desviando destes objetos até chegar em seu destino final.

REFERÊNCIAS

ABRÓSIO, J. A. C. **Advances in Computational Multibody Systems.** 1. ed. Netherlands: Springer, 2005.

ARDUINO Mega 2560. Disponivel em: <http://arduino.cc/en/Main/arduinoBoardMega2560>. Acesso em: 20 set. 2014.

BEKEY, G.; YUH, J. **The Status of Robotics. Report on the WTEC International Study: Part II**. IEEE Robotics and Automation Magazine, p. v. 15, n. 1, p. 80-86. 2008.

BORGES, A. A. F. **Proposta de Navegação Segura com Sistema de Controle Embarcado Aplicado em Plataformas Robóticos Móveis.** UEL. Londrina. 2014.

CANUDAS DE WIT, C.; SICILIANO, B.; BASTIN, G. **Theory of Robot Control**. Springer. 1997.

CARELLI, R.; SECCHI, H.; MUT, V. **Algorithms for Stable Control of Mobile Robots with Obstacle Avoidance**. Latin American Applied Research, v. 29, n. 3/4, p. 191-196. 1999.

CONTROL Tutorials for MatLab & Simulink. Disponivel em: <http://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SystemModeling>. Acesso em: 09 dez. 2014.

CRUZ, R. Estadão. **blogs.estadao.com.br/**, 2014. Disponivel em: <http://blogs.estadao.com.br/renato-cruz/a-invasao-dos-robos/>. Acesso em: 09 out. 2014.

DE LA CRUZ, C.; CARELLI, R. **Dynamic Modeling and Centralized Formation Control of Mobile Robots.** In: 32nd IEEE Conference on Industrial Electronics, p. 3880-3885. 2006.

DORF, C. R.; ROBERT, B. H. **Sistemas de Controle Moderno**. 8. ed. Rio de Janeiro: LTC , 2001.

DUDEK, G.; JENKIN, M. **Computational Principles of Mobile Robotics**. Cambridge CB2 2RU, UK: Press Syndicate of the University of Cambridge. 2000.

FIERRO, R.; DAS, A. **A Modular Architecture for Formation Control**. Third International Workshop on Robot Motion and Control. [S.l.], p. 285-290. 2002.

FIERRO, R.; LEWIS, F. L. **Control of a Nonholonomic Mobile Robot**. Backstepping Kinematics into Dynamics. Journal of Robotic Systems, p. v. 14, n. 3, p. 149-163. 1997.

FRAGA, S.; SOUSA, J.; PEREIRA, F. **Geração de Trajetórias para Sistemas Diferencialmente Planos**. III Festival Nacional de Robótica - ROBOTICA2003. Lisboa, Portugal. 2003.

GROOVER, P. M. **Automação Industrial e Sistema de Manufatura**. 3. ed. São Paulo: Pearson Prentice Hall, 2011.

KANAYAMA, Y. E. A. **A Stable Tracking Control Method for an Autonomous Mobile Robot**. IEEE International Conference on Robotics and Automation, p. p. 384-389. 1990.

KIM, M.; SHIN, J.; LEE, J. **Design of a Robust Adaptive Controller for a Mobile Robot**. Proc. of the IEEE/RSJ Inst. Conf. on Intelligent Robots and Systems. [S.l.], p. v. 3, p. 1816-1821. 2000.

LIMA, B. C.; MARCO, V. M. V. M. **AVR e Arduino Técnicas de Projeto**. 2. ed. Florianópolis: Clube de Autores, 2012.

MARTINS, F. N. **et al. Dynamic Modeling and Adaptive Dynamic Compensation for Unicycle-Like Mobile Robots**. 14th International Conference on Advanced Robotics - ICAR 2009. Munique, Alemanha, 22 a 26 de Junho 2009. 2009.

MARTINS, F. N. **A Multi-Layer Control Scheme for Multi-Robot Formations with Adaptive Dynamic Compensation**. In: XIII Reunión de Trabajo en Procesamiento de la Información y Control - RPIC. Rosario, Argentina. 2009.

MATHWORKS. **MatLAb Central**, 2014. Disponivel em: <http://www.mathworks.com/matlabcentral/fileexchange/32374-matlab-support-for-arduino--aka-arduinoio-package->. Acesso em: 10 dez. 2014.

MELLO, F. L. **Proposta de Simulador Virtual para Sistemas de Navegação de Robôs Móveis Utilizando Conceitos de Prototipagem Rápida**. UNICAMP. Campinas. 2007.

NIKU, B. S. **Introdução à Robótica - Análise, Controle, Aplicações**. 2. ed. Rio de Janeiro: LTC, 2013.

NISE, S. N. **Engenharia de Sistemas de Controle**. 6. ed. Rio de Janeiro: LTC, 2012.

PAGNOTELLI, S.; VALIGI, P. S. A flexible framework for rapid prototyping of mobile robotics applications. **14th Mediterranean Conference on Control and Automation**, USA, 2006. p.1-5.

ROMANO, V. F. **Robótica Industrial:** Aplicação na Indústria de Manufatura e de Processos. [S.l.]: Edgard Blucher Ltda, 2002.

SIEGWART, R.; NOURBAKHSH, I. R. **Introduction to Autonomous Mobile Robot Intelligent Robotics and Autonomous Agents**. The MIT Press. Massachusetts Institute of Technology. 2004.

SPONG, M.; HUTCHINSOS, S.; VIDYASAGAR, M. **Robot Modeling and Control**. John Wiley and Sons, 2006: [s.n.], 2006.

APÊNDICE A – DISSEMINAÇÕES

A.1 Publicações relacionadas com o tema

- Melo, L.F.,Rosário, J. M., Rodrigues, E. J. **Wheelchair Secure Navigation With RF Signal Triangulation and Genetic Algorithm Optimization**. IGI Glogal Publisher. USA. 2014

APÊNDICE B – LISTAGEM DA ROTINA DO MODELO DINÂMICO DO ROBÔ

A rotina de geração do modelo dinâmico do robô móvel foi desenvolvida em linguagem script do MatLab e tem como base matemática as equações desenvolvidas no capítulo 3 nas seções 3.4.5 e 3.4.6.

```
% Inicializacao do controlador do motor DC
clear;
clc;

Ra=7.2;
La=3400E-6;
Kt=13.15e-3;
Kb=13.178E-3;
Ka=1;
Jm=4.5E-7;
b=4.7989e-6;
Omegamax=984.36;
N=1/84;
Kenc=7.63;
Kadc=4.88E-3;

% Montagem das matrizes e a equação de estado
A=[-Ra/La 0 -Kb/La; 0 0 1; Kt/Jm 0 -b/Jm]
B=[(Ka*Kadc)/La;0;0]
%C=[0 Kenc 0]
C=[0 0 N]
D=[0]

% Transformando a equação de estado em FT.
[num,den]=ss2tf(A,B,C,D)

%Sistema em Malha Fechada Acima:
numSF=num
denSF=(den+num)

%Controladores
Kp = 149.8811;
Ki = 29532.3607;
numSF = [Kp Ki];
denSF = [1 0];
[num2, den2] = cloop(conv(num, numSF), conv(den, denSF),-1)

%Discretização do Sistema
Ts=1;
[num3 den3] = c2dm(num2, den2,Ts,'zoh')
printsys(num3,den3,'z')
```

APÊNDICE C – LISTAGEM DA ROTINA DO MODELO CINEMÁTICO DO ROBÔ

A listagem da rotina de geração do modelo cinemático do robô móvel foi elaborada no MatLab e tem como base matemática as equações desenvolvidas no capítulo 3 nas seções 3.1, 3.2 e 3.3.

```
function [X,Y]  = SimulaTrajetoriaRobo(V_d2,V_e2,t)

hold on;
%Posição Inicial do robô
X_i = 0;
Y_i = 0;
Teta_i = 0;
L = 280;

%Fórmula Desenvolvida: Vreal = Sinal de Entrada × RPM × 0.10472
%Sinal de Entrada = V_d2 e V_e2
% V_d2 = %Velocidade Roda Direita do Robô Real (mm/s)
% V_e2 = %Velocidade Roda Esquerda do Robô Real (mm/s)
    t1=t;
    X = 0;
    Y = 0;
    Teta_f =  0

for i=t

    %trajetória reta
    if V_d2 == V_e2
    V = V_d2;
    %X = vetor posição do robo no eixo X
    X = X_i + V * cos(Teta_i) .* t;
    %Y = vetor posição do robo no eixo Y
    Y = Y_i + V * sin(Teta_i) .* t;
    Teta_f = Teta_i;
else                %trajetória circular
    %R_c = raio de curvatura do robô
    R_c = (L * (V_d2 + V_e2)) / (2 * (V_d2 - V_e2));
    %W = velocidade angular do robô em rad/s
    W = (V_d2 - V_e2) / L;
    %Teta = deslocamento angular do robo no tempo em rad
    Teta = Teta_i + W .* t;
    %X = vetor posição do robo no eixo X
    X = X_i - R_c * sin(Teta_i) + R_c .* sin(Teta);
    %Y = vetor posição do robo no eixo Y
    Y= Y_i - R_c * (1 - cos(Teta_i)) + R_c .* (1 - cos(Teta));
    [nT,mT] = size(Teta);
    Teta_f = Teta(mT);
   Teta_f = Teta;
    Cx = X(1) - R_c(1)*sin(Teta(1));
    Cy = Y(1) + R_c(1)*cos(Teta(1));

   end
end
```

```
a=plot(X,Y,'LineWidth',2,'Marker','o','Color',[0.84,0.16,0]);
title('\bfTrajetória Real do Robô - Reta');
xlabel('X (mm)');
ylabel('Y (mm)');
hold on;
;
```

APÊNDICE D - LISTAGEM DA ROTINA DO GERADOR DE TRAJETÓRIAS

Nesta seção mostra-se a lista de rotina para a geração de trajetórias, também elaborada no MatLab, e tem como objetivo visualizar as trajetórias que são geradas de acordo com as velocidades angulares configuradas para cada roda do robô, representadas pelas variáveis Vrpm1, Vrpm1_2, Vrpm2, Vrpm2_2, Vrpm3,Vrpm3_2, onde por exemplo Vrpm1 representa a velocidade da roda direita do robô e Vrpm1_2 a velocidade da roda esquerda, sendo cada para configurado para emitir uma velocidade durante um intervalo de tempo.

```
%Script Gerador de Trajetórias

%Velocidades em RPM - Entre com os valores para gerar a trajetória.
Vrpm1 = 45;
Vrpm1_2 = 85;

Vrpm2 = 78;
Vrpm2_2 = 55;

Vrpm3 = 45;
Vrpm3_2 = 85;

%Fórmula para converter a velocidade de RPM para mm/s
V_d1 = (Vrpm1 *49.35*0.10472)/2;
V_e1 = (Vrpm1_2 *49.35*0.10472)/2;

V_d2 = (Vrpm2 *49.35*0.10472)/2;
V_e2 = (Vrpm2_2 *49.35*0.10472)/2;

V_d3 = (Vrpm3 *49.35*0.10472)/2;
V_e3 = (Vrpm3_2 *49.35*0.10472)/2;

X_i = 0;    %posição inicial
Y_i = 0;
Teta_i = 0;
L = 280;    %distancia entre as rodas em mm

intervalo1 = 4;  %Numero de colunas a ser criado para o vetor tempo
intervalo2 = 14;
intervalo3 = 4;

%Vetor Tempo(vai criar um vetor para contar o tempo(cada coluna = 1s)
Vetor_Tempo1 = 1;
t1=0:Vetor_Tempo1:intervalo1; %vetor tempo em s

Vetor_Tempo2 = 1;
t2=0:Vetor_Tempo2:intervalo2; %vetor tempo em s

Vetor_Tempo3 = 1;
t3=0:Vetor_Tempo3:intervalo3; %vetor tempo em s
```

```
[X1,Y1,Teta_i2] = trajetoria(V_d1,V_e1,X_i,Y_i,Teta_i,L,t1);

[~,mx] = size(X1);      %X1(mx) = último elemento do vetor X1
X_i2 = X1(mx);
[~,my] = size(Y1);
Y_i2 = Y1(my);

[X2,Y2,Teta_i3] = trajetoria(V_d2,V_e2,X_i2,Y_i2,Teta_i2,L,t2);

[~,mx] = size(X2);      %X1(mx) = último elemento do vetor X1
X_i3 = X2(mx);
[~,my] = size(Y2);
Y_i3 = Y2(my);

[X3,Y3,Teta_i4] = trajetoria(V_d3,V_e3,X_i3,Y_i3,Teta_i3,L,t3);

[nx,mx] = size(X3);      %X1(mx) = último elemento do vetor X1
X_i4 = X3(mx);
[ny,my] = size(Y3);
Y_i4 = Y3(my);

%os vetores X e Y finais são a concatenação de X1 e X2, X3.
X = [X1,X2,X3];  %cat(X1,X2,X3);
Y = [Y1,Y2,Y3];  %cat(Y1,Y2,Y3);
Tt= [t1,t2,t3];

clf; %Vai gerar um quadro
figure(1); %Vai escrever neste quadro Figure 1
hold on; %Gera uma tela em branco eixo X e Y
plot(X,Y,'bs:','lineWidth',2);%Gera o gráfico
title('\bfTrajetória do robô móvel no plano XY - CIRCULO RAIO 1M');%Coloca
um título no gráfico
xlabel('Posição (mm)');%No eixo X escreve Posição (mm)
ylabel('Posição (mm)');%No eixo X escreve Posição (mm)
axis equal;%Da um zoom sobre o gráfico
hold off;%Abondona a tela
```

APÊNDICE E - LISTAGEM DA ROTINA DO VISUALIZADOR DE TRAJETÓRIAS REAIS

Esta lista de rotina foi elaborada utilizando o MatLab com o objetivo de visualizar através de um gráfico a trajetória percorrida pelo robô. Este script traça o gráfico com base nos dados armazenados em uma planilha de excel, a quantidade de pulsos gerados por segundo pelo encoder, em relação a cada roda do robô, conforme explicado na seção 5.5.3.

```
%Script para visualização da trajetória percorrida pelo robô.
hold on;

%parâmetros do robô
X_i = 0;    %posição inicial
Y_i = 0;
Teta_i = 0;
L = 280;    %distancia entre as rodas em mm

%v = r × RPM × 0.10472
Pulsos=csvread('Dados.csv',1,0);

%Transforma Pulsos em Vrpm
Vrpm_a1 = 60*(1/96)*(1/84)
Vrpm_b1 = 60*(1/96)*(1/84)

%Transforma Vrpm em Vmm/s
va1=Pulsos(:,1)*24.67*0.10472*Vrpm_a1);
vb1=Pulsos(:,2)*24.67*0.10472*Vrpm_b1);
t1=Pulsos(:,3);

% ler os dados da trajetória quando o robô está no chão
for i=1:size(va1,1)

    [XP(i),YP(i),Teta_f] = trajetoria(va1(i),vb1(i),X_i,Y_i,Teta_i,L,i);
end
a=plot(XP,YP,'LineWidth',2,'Marker','o','Color',[0.70,0.12,0]);
hleg1 = legend('Trajetória Real');
hold on;
```

APÊNDICE F - LISTAGEM DA ROTINA DO COMPARADOR DE TRAJETÓRIAS

Este Script elabora no MatLab, tem a finalidade de comparar a trajetória simulada com a trajetória real, já explicado na seção 6.1.

```
%Scipt para comparar a trajetória simulada com a real.
run Gerador_Trajetoria;
hold on;

pulsos=csvread('Dados_Reta.csv',1,0);

%parâmetros do robô
X_i = 0;    %posição inicial
Y_i = 0;
Teta_i = 0;
L = 280;    %distancia entre as rodas em mm

%Transforma Pulsos em Vrpm
Vrpm_a1 = 60*(1/96)*(1/84)
Vrpm_b1 = 60*(1/96)*(1/84)

%Transforma Vrpm em Vmm/s -- v = (d × RPM × 0.10472)/2
va1=pulsos(:,1)*24.67*0.10472*Vrpm_a1);
vb1=pulsos(:,2)*24.67*0.10472*Vrpm_b1);
t1=pulsos(:,3);

for i=1:size(va1,1)

        [XP(i),YP(i),Teta_f] =
trajetoria(va1(i),vb1(i),X_i,Y_i,Teta_i,L,i);

end
a=plot(XP,YP,'LineWidth',2,'Marker','o','Color',[0.70,0.12,0]);
hleg1 = legend('Trajetória Real');
hold on;
```

APÊNDICE G – SKETCH PARA GRAVAÇÃO DOS DADOS DO ENCODER NO SD

Este algoritmo foi elabora no próprio software do Arduino e são conhecidos como Sketch, tendo a finalidade de elaborar pequenos códigos para serem compilados para o microcontrolador. O propósito deste código já foi dito na seção 5.5.2.

```
#define ETHSDON
#define SERIALRCV
//#define TIMELIMIT

#ifdef ETHSDON
#include <SD.h>
#endif ETHSDON

#include <TimerOne.h>
#include <math.h>

// Static variables
static char dtostrfbuffer[15];
// Standard variables
volatile word timeSecs;
volatile word numberOfPulsesA, numberOfPulsesB;
volatile word velocityRPMA,velocityRPMB;
volatile boolean serialTXFree;
// Controller Variables
float SetpointA, InputA, ErrorA, OutputA;
float SetpointB, InputB, ErrorB, OutputB;
float kp=1.0, ki=1, kd=1;

// Ethernel SD Shield specific variables
#ifdef ETHSDON
const int chipSelect = 53;
String dataString;
boolean firstRun=true;
#endif ETHSDON

void setup()
{
  // Misc configurations and serial port enable
  pinMode(2, INPUT);
  Serial.begin(115200);
  // enable pins activate all
  digitalWrite(22, HIGH);
  digitalWrite(24, HIGH);
  digitalWrite(26, HIGH);
  // Timer interrupts setuo
  attachInterrupt(0, isr_ext0, HIGH);
  attachInterrupt(1, isr_ext1, HIGH);
  Timer1.initialize(1000000);
  Timer1.attachInterrupt(Timer1ISR);

  // Ethernet-SD Shield specific setup
  #ifdef ETHSDON
  // Start SD Card
  Serial.print("Initializing SD card...");
```

```
  // make sure that the default chip select pin is set to
  // output, even if you don't use it:
  pinMode(53, OUTPUT);
  // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect))
  {
    Serial.println("Card failed, or not present");
    // don't do anything more:
    while(1);
  }
  Serial.println("card initialized.");
  // Enable TimerOne and external interrupts
  #endif ETHSDON
}

void loop()
{
  // EVASIVE MANUEVER PARAMETERS ACCORDING TO MATLAB
  float kspd=1.2;
  if(timeSecs<4)
  {
    OutputA=198;
    OutputB=196.8;
  }
  if(timeSecs>4 && timeSecs<=8)
  {
    OutputA=198;
    OutputB=196.8;
  }
  if(timeSecs>8 && timeSecs<=12)
  {
    OutputA=198;
    OutputB=196.8;
  }
  if(timeSecs>12 && timeSecs<=19)
  {
    OutputA=198;
    OutputB=196.8;
  }
  if(timeSecs>19)
  {
    analogWrite(8, 255);
    analogWrite(9, 255);
    digitalWrite(22, LOW);
    digitalWrite(24, LOW);
    while(1);
  }
  // Motor velocity simple proportional control block

  if(OutputA<0)
  {
    OutputA=0;
  }
  if(OutputB<0)
  {
    OutputB=0;
  }
  if(OutputA>255)
  {
    OutputA=255;
  }
```

```
if(OutputB>255)
{
  OutputB=255;
}
analogWrite(8, OutputA);
analogWrite(9, OutputB);
// Serial data
if(serialTXFree)
{
  //Serial.print("|RPM|");
  Serial.print(velocityRPMA);
  Serial.print(",");
  Serial.print(velocityRPMB);
  //Serial.print("|Output|");
  //Serial.print(OutputA);
  //Serial.print(",");
  //Serial.print(OutputB);
  //Serial.print("|Input|");
  //Serial.print(InputA);
  Serial.print(",");
  //Serial.print(InputB);
  //Serial.print("||");
  Serial.println(timeSecs);
}
serialTXFree=0;
// Ethernet-SD Shield specific commands
#ifdef ETHSDON
// SD Card write sequence
File dataFile = SD.open("teste2.txt", FILE_WRITE);
// if the file is available, write to it:
if (dataFile)
{
  if(firstRun)
  {
    dataFile.println("VRPMA,VRPMB,TimeSecs");
    firstRun=false;
  }
  dtostrf(velocityRPMA,8, 2, dtostrfbuffer);
  dataFile.print(dtostrfbuffer);
  dataFile.print(",");
  dtostrf(velocityRPMB,8, 2, dtostrfbuffer);
  dataFile.print(dtostrfbuffer);
  dataFile.print(",");
  dtostrf(timeSecs,8, 2, dtostrfbuffer);
  dataFile.println(dtostrfbuffer);
  dataFile.close();
}
// if the file isn't open, pop up an error:
else
{
  Serial.println("error opening teste2.txt");
  digitalWrite(13, HIGH);
  return;
  while(1);
}
#endif ETHSDON
// Time limit variable
#ifdef TIMELIMIT
if(timeSecs==10)
{
  analogWrite(8, 255);
```

```
    analogWrite(9, 255);
    digitalWrite(22, LOW);
    digitalWrite(24, LOW);
    while(1);
  }
  #endif TIMELIMIT
}

void isr_ext0()
{
  numberOfPulsesA++;
}

void isr_ext1()
{
  numberOfPulsesB++;
}

void Timer1ISR()
{
  velocityRPMA=numberOfPulsesA;
  velocityRPMB=numberOfPulsesB;
  //velocityRPMA=((numberOfPulsesA/96)*60)/84;
  //velocityRPMB=((numberOfPulsesB/96)*60)/84;
  numberOfPulsesA=0;
  numberOfPulsesB=0;
  timeSecs++;
  serialTXFree=1;
}
```

APÊNDICE H – IMAGENS EM 3D E FOTOS DO ROBÔ MÓVEL

Neste Apêndice mostra-se na Figura H.1 o projeto em 3D do robô móvel e na Figura H.2 a foto real do robô móvel junto a seus componentes com suas respectivas descrições. Desta forma na Figura H.1 apresenta as imagens em 3D em diferentes ângulos da visão do robô móvel

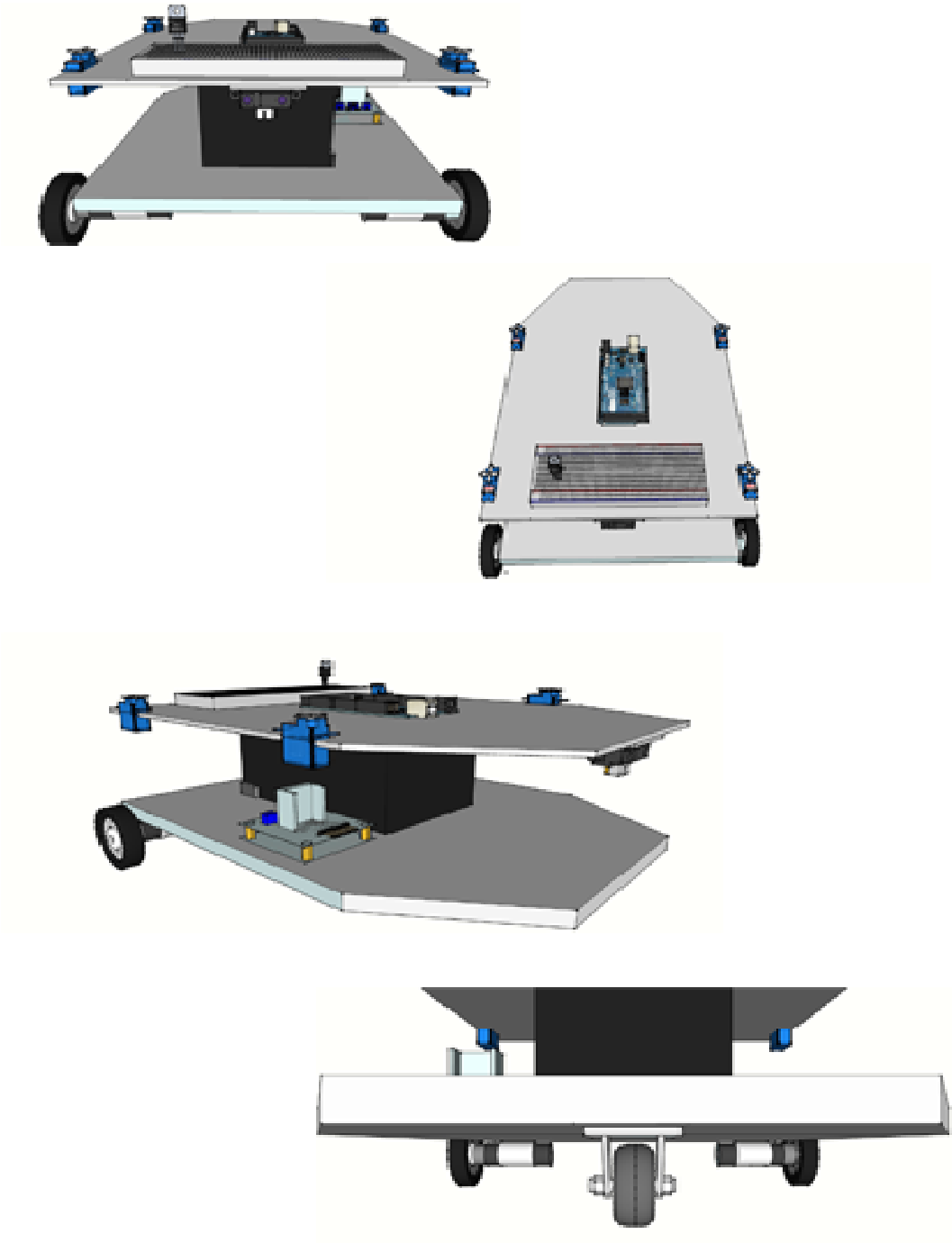


Figura H.1: Imagens em 3D do Robô Móvel

Na Figura H.2 mostra-se a foto tirada do robô móvel, ilustrando-se seus componentes.

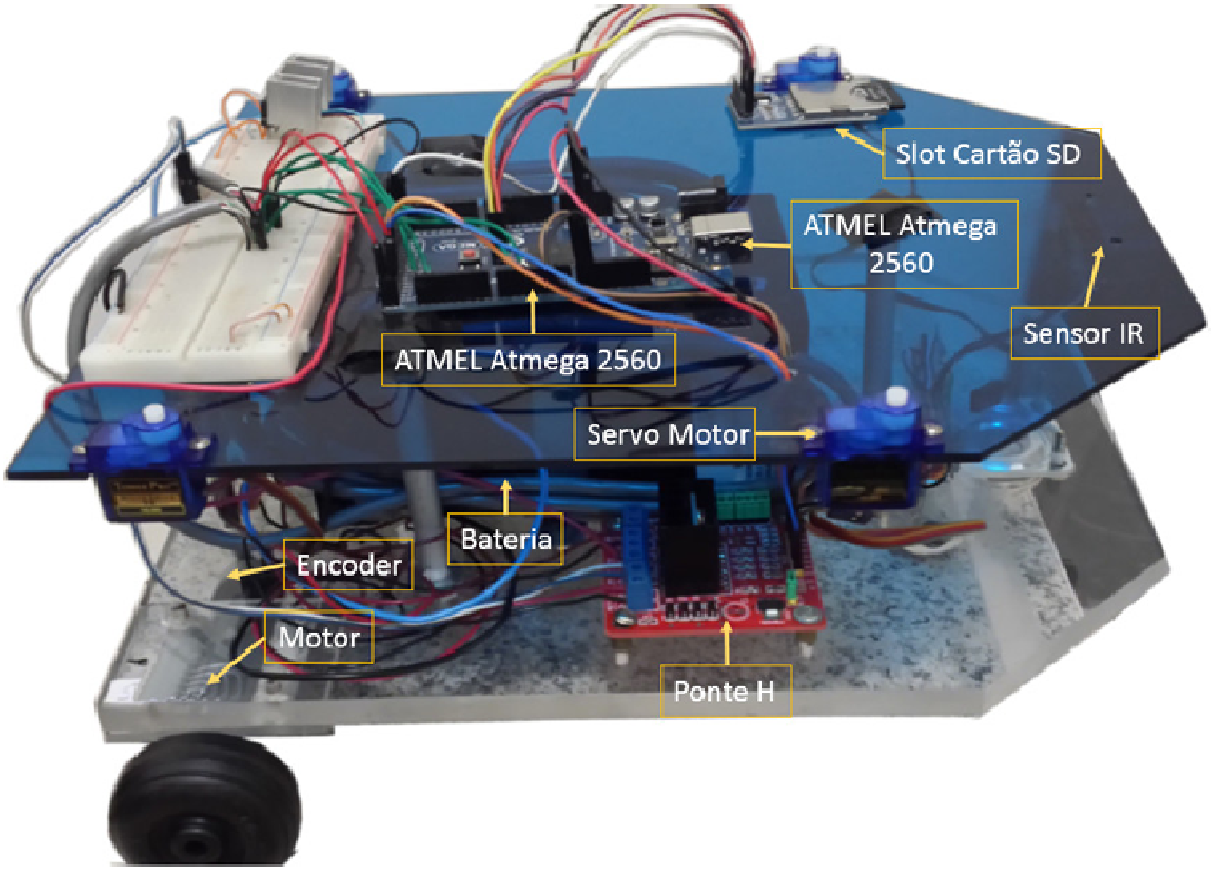
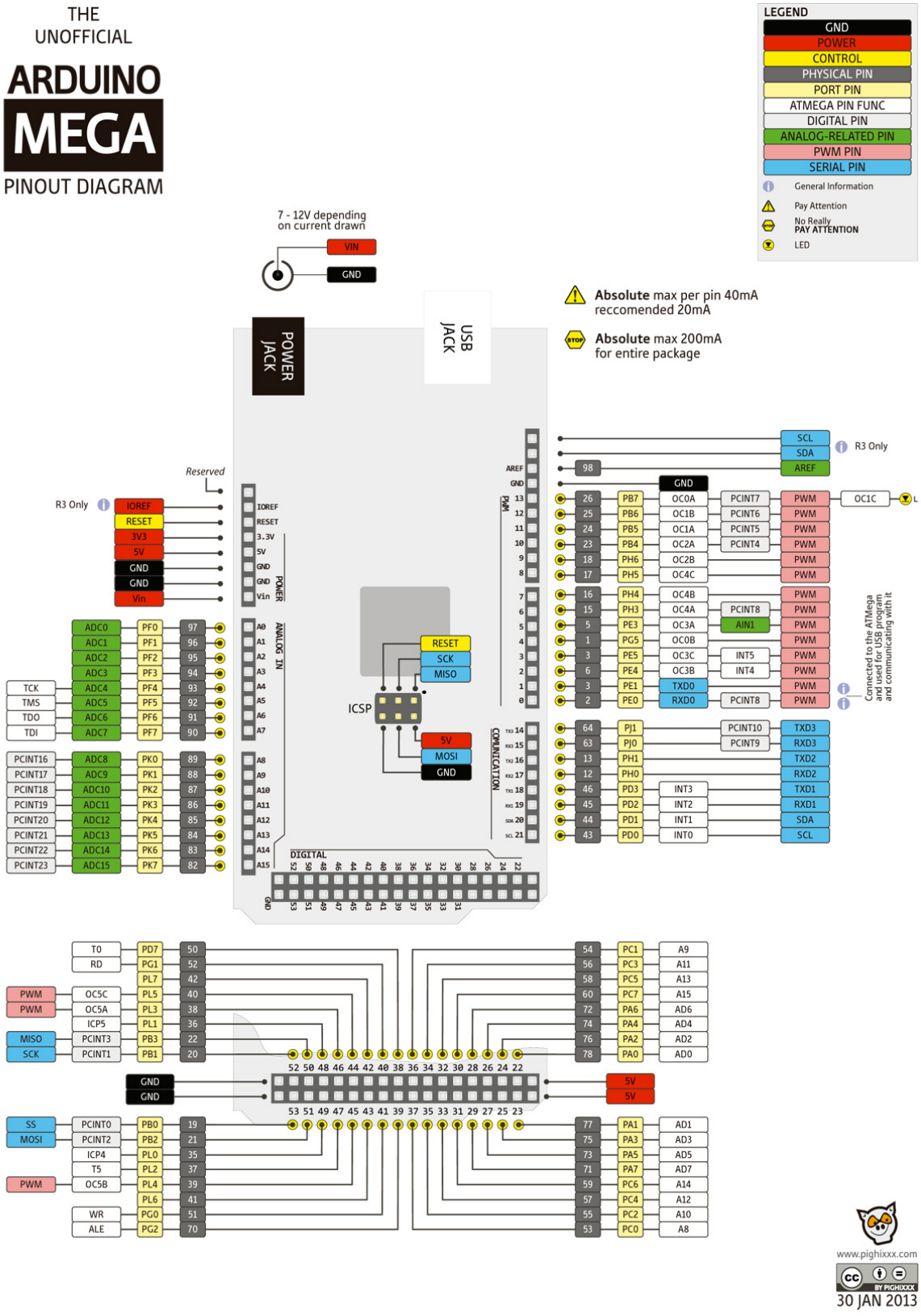


Figura H.2: Foto do robô móvel com seus acessórios apresentados

ANEXO A - MICROCONTROLADOR ATMEGA 2560



ANEXO B - DADOS DO FABRICANTE - ENCODER

No Anexo B é exibido o relato do encoder AMT103 do fabricante CUI INC utilizado neste trabalho.

CUI INC

date 09/2009
page 1 of 5

SERIES: M223X000X **DESCRIPTION: M223 with AMT103 kit**

AMT SPECIFICATIONS

output phase difference	90° ±45°
frequency response	0 ~ 250 kHz
output current	0 ~ 5 mA max.
output waveform	square wave
output signals	A, B, Z phase
current consumption	6 mA typ., 10 mA max.
supply voltage	3.6 ~ 5.5 V dc
output resolution (ppr)[1]	48, 96, 100, 125, 192, 200, 250, 256, 384, 400, 500, 512, 800, 1000, 1024, 2048
time constant	0.2 ms (at 48, 96, 100, 125, 200, 250, 256, 512 ppr) 0.4 ms (at 192, 384, 400, 500, 800, 1000, 1024, 2048 ppr)
accuracy	±15 arcmin (at 192, 384, 400, 500, 800, 1000, 1024, 2048 ppr) ±30 arcmin (at 96, 200, 250, 512 ppr) ±60 arcmin (at 48, 100, 125, 256 ppr)
max. rotational speed	7500 rpm (at 384, 800, 1000, 2048 ppr) 15000 rpm (at 192, 400, 500, 1024 ppr) 30000 rpm (at 48, 96, 100, 125, 200, 250, 256, 512 ppr)
operating temp.	-40° ~ 100°C
mounting hole options	A) 2 each M1.6 holes on 16 mm (0.63") bolt circle B) 2 each #4 holes on 19.05 mm (0.75") bolt circle C) 2 each M1.6 or M2 holes on 20 mm (0.787") bolt circle D) 3 each M1.6 or M2 holes on 20.9 mm (0.823") bolt circle E) 3 each M1.6 or M2 holes on 22 mm (0.866") bolt circle F) 4 each M1.6 or M2 holes on 25.4 mm (1") bolt circle

note: 1. All resolutions stated are before quadrature decoding (example: 1000 ppr x 4 = 4000 counts)
2. Some stepper motors may leak a magnetic field causing the AMT index pulse to not function properly.

ANEXO C - DADOS DO FABRICANTE MOTOR CC

Nesta seção é apresentado a descrição do motor M233 do fabricante CUI INC utilizado neste trabalho.

SERIES: M223X000X **DESCRIPTION: M223 with AMT103 kit**

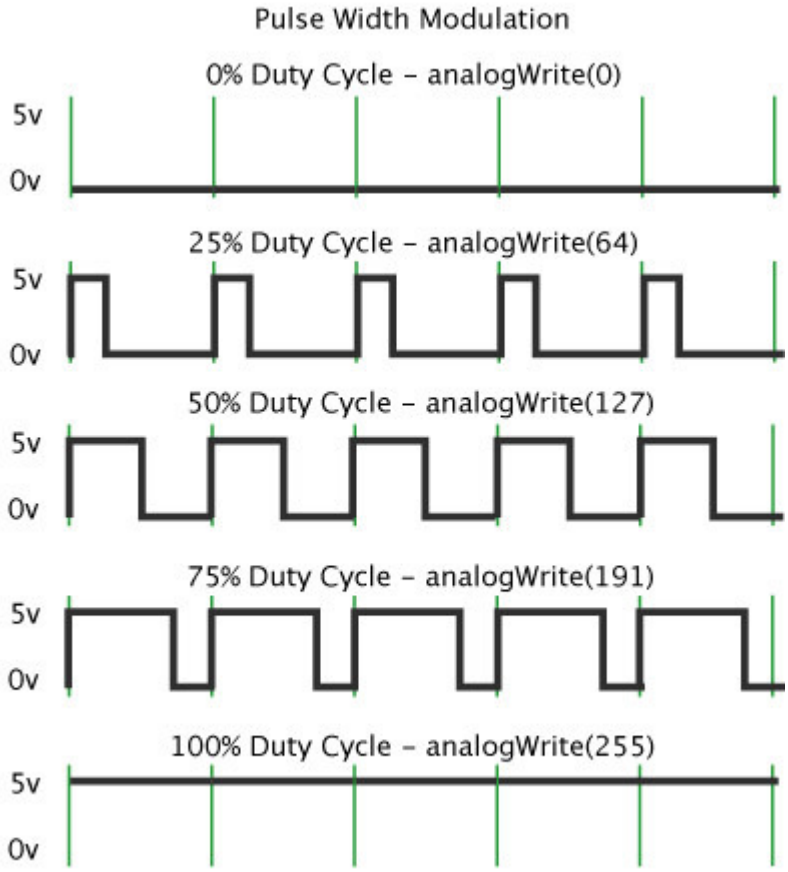
M223X DC MOTOR SPECIFICATIONS

Micro-Drives

parameter	conditions/description	min	nom	max	units
gear reduction	M223X0002 M223X0003 M223X0004		none 1:84 1:231		
voltage			12		V dc
winding resistance		6.34	7.2	8.06	Ω
output power				4.65	W
efficiency				66	%
no load speed		7,400	8,400	9,400	rpm
no load current	shaft ø2 mm	0.03	0.06	0.09	A
stall torque			21.92		mNm
friction torque			0.79		mNm
speed constant			726		rpm/V
back-EMF constant			1.38		mV/rpm
torque constant			13.15		mNm/A
current constant			0.076		A/mNm
rotor inductance			3,400		µH
mechanical time constant			19		ms
rotor inertia			4.5		gcm²
angular acceleration				49	10^3 rad/s²
thermal resistance			17/22		k/W
thermal time constant			450		s
operating temp.		-10		50	°C
shaft bearing	sintered bronze sleeves				
shaft load	shaft ø2 mm: radial at 3,000 rpm (3 mm from bearing) axial at 3,000 rpm axial at standstill (shaft supported)			 2 1 294	 N N N
shaft play	radial ≤ axial ≤	 0.5		0.025 0.45	mm mm
direction of rotation	clockwise, viewed from the front face				
speed up to				10,000	rpm
torque up to				5,307	mNm
current up to	thermal limits			0.502	A

ANEXO D – FUNCIONAMENTO DO PWM

Neste anexo mostra-se o funcionamento do PWM utilizado para controlar os motores CC.



Fonte: http://arduino.cc/en/Tutorial/PWM