UNIVERSIDADE LUTERANA DO BRASIL

PRÓ-REITORIA DE GRADUAÇÃO

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

EVANDRO LOCATELLI

CONTROLE DE VELOCIDADE PARA VEÍCULO AUTÔNOMO

Canoas, Dezembro 2009

EVANDRO LOCATELLI

CONTROLE DE VELOCIDADE PARA VEÍCULO AUTÔNOMO

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da ULBRA como um dos requisitos obrigatórios para a obtenção do grau de Engenheiro Eletricista

Departamento:

Engenharia Elétrica

Área de Concentração

Robótica, Eletrônica Embarcada

Professor Orientador:

MSc. Eng. Eletr. Augusto Alexandre Durgante de Mattos– CREA - RS 88.003

Canoas

2009

FOLHA DE APROVAÇÃO

Nome do Autor: Evandro Locatelli

Matrícula: 0210099089-9

Título: Controle de Velocidade para Veículo Autônomo

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da ULBRA como um dos requisitos obrigatórios para a obtenção do grau de Engenheiro Eletricista

Professor Orientador:

MSc. Eng. Eletr. Augusto Alexandre Durgante de Mattos

CREA: RS088003-D

Banca Avaliadora:

MSc. Eng. Eletr. Dalton Luiz Rech Vidor

CREA: RS079005-D

Conceito Atribuído (A-B-C-D):

MSc. Eng. Eletr. Miriam Noemi Cáceres Villamayor

CREA- RS067231-D

Conceito Atribuído (A-B-C-D):

Assinaturas:

Autor Evandro Locatelli	Orientador Augusto Alexandre Durgante de Mattos
Avaliador Dalton Luiz Rech Vidor	Avaliador Miriam Noemi Cáceres Villamayor

Relatório Aprovado em:

DEDICATÓRIA

Dedico a aos meus pais aos meus irmãos e a minha noiva Aline.

AGRADECIMENTOS

A minha formação como profissional não poderia ter sido concretizada sem a ajuda de meus amáveis e eternos pais, Baldino e Gloria, que, no decorrer da minha vida, proporcionaram-me, além de extenso carinho e amor, os conhecimentos da integridade.

Aos meus irmãos Évertom e Cassiano, as cunhadas Gisele e Valéria.

Ao meu sobrinho Tiago que por inúmeras vezes pediu que eu o deixasse brincar com o "carrinho".

A minha noiva Aline que permaneceu sempre ao meu lado nos bons e maus momentos que além de me fazer feliz, ajudou-me durante todo o percurso.

A nona Linda, *in memoriam*, que em muitas ocasiões não pude estar presente por me dedicar aos estudos.

Ao professore Augusto pelos conselhos e pela dedicação na orientação deste trabalho e aos professores Cocian, Dalton, Godoy, Marilia, Mirinam e Vernetti pelo ensinamento passado durante as aulas.

Aos ex. colegas e agora engenheiros Rodrigo Brandit, Rodrigo Bolico, Rogério Volcan, Flavio Giordani, e ao que ainda será, Emerson Pedrotti.

À todos vocês, meu muito obrigado.

EPÍGRAFE

Se há mais de um modo de se fazer um trabalho, e um desses modos resultará em desastre, então alguém o fará.

Edward Murphy.

RESUMO

LOCATELLI, Evandro, **Controle de Velocidade para Veículos Autônomos**. Trabalho de Conclusão de Curso em Engenharia Elétrica - Departamento de Engenharia Elétrica. Universidade Luterana do Brasil. Canoas, RS. 2009.

Este trabalho descreve o desenvolvimento de um sistema de controle de velocidade para veículos autônomos onde foi concebido um protótipo funcional. Em sua estrutura foram utilizados dois motores CC acoplados nas rodas dianteiras, sendo acionados pela técnica de *Pulse Width Modulation* (PWM), a estratégia de controle utilizada foi em malha fechada com um controlador proporcional integral - PI independente para cada roda. O *feedback* de velocidade das rodas é através de encoders, as rodas dentadas foram acopladas nos eixos dos motores e o sistema óptico aciona as interrupções do micro-controlador. O sistema de processamento é implementado com o micro-controlador da família PIC 18F que é programado com linguagem de programação compilada de alto nível, em C.

Palavras chave: Veículos Autônomos. Sistema de Controle. Programação. Motores.

ABSTRACT

LOCATELLI, Evandro, Speed Control for Autonomous Vehicles. Conclusion Course in Electrical Engineering - Department of Electrical Engineering. Lutheran University of Brazil. Canoas, RS. 2009.

This paper describes the development of a system of speed control for autonomous vehicles to be implemented in a prototype. In this prototype was used two DC motors coupled to the front wheels being driven by the technique of Pulse Width Modulation (PWM), the control strategy in closed loop was a proportional integral controller (PI) independently for each wheel. The feedback from the wheel speed is detained by encoders, where the optical discs were engaged in the axes of the engines. The pulse width is measured using the interrupt in the micro-controller. The processing system was implemented with the micro-controller PIC 18F family programmed with the programming language compiled high level in C.

Keywords: Vehicle Autonomus. Sistem of control. Programming. Motors.

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE FÍGURAS

LISTA DE ABREVIATURAS E SIGLAS

AGV - *Automatic Guided Vehicle;*

BLDC - *Brushless* DC;

CA - Corrente alternada;

CC - Corrente continua;

CI - Circuito Integrado;

erp - Erro em regime permanente;

erro_md - Erro de velocidade motor da direita do motor da direita;

fc - Frequência de corte;

FCEM - Força Contra-Eletromotriz;

FTS - *Fahrerlose Transportsysteme;*

ie - Corrente do emissor;

ir - Corrente do receptor;

IR - *Infrared*;

kd - Constante da ação derivativa;

ki - Constante da ação integral;

ki_md - Parcela da ação integral – I do motor da direita;

ki_term_ant_md - Valor anterior da ação integral do motor da direita;

ki_term_md - Valor atual da ação integral do motor da direita;

kp - Constante da ação proporcional;

kp_md - Parcela da ação proporcional - P do motor da direita;

LCD - *liquid crystal display*;

LDR - *Light Dependent Resistor;*

LGV - *Laser Guided Vehicle;*

MV - *Variável manipulada*;

PPR - Pulso por Resolução;

PV - *Variável de processo*;

PWM - *Pulse Width Modulation* (Modulação por Largura de Pulso);

sp_velocidade_md - *Set point* de velocidade do motor da direita;

t - Tempo integral;

t1 - Tempo do pulso desligado;

t1 - Tempo do pulso ligado;

Ts - Tempo de estabilização;

V - Tensão de alimentação;

ve - Tensão do emissor;

velocidade_md - Velocidade instantânea do motor da direita;

vr - Tensão do receptor;

LISTA DE UNIDADES

s - segundos;

ms - mili segundos;

m - metros;

cm - centímetros;

m/s - metros por segundo;

m/min - metros por minuto.

SUMÁRIO

1. INTRODUÇÃO

O interesse ou necessidade de controlar, de forma autônoma, um veiculo vem sendo discutido e estudado por pesquisadores e estudantes das mais diversas áreas de tecnologia. A aplicação para os *Automatic Guided Vehicle* (AGV) pode ser encontrada nas mais diversas áreas, como, comercial, fabricas, automatizando armazéns, na locomoção ou inspeção de ambientes insalubres. O AGV também pode ser chamado *Laser Guided Vehicle* (LGV). Na Alemanha, a tecnologia também é chamada *Fahrerlose Transportsysteme* (FTS).

O primeiro AGV foi implantado no mercado na década de 1950, por *Barrett* Eletrônica de *Northbrook*, na ocasião foi simplesmente um reboque que seguia um fio, instalado no chão, que substituiu uma via férrea. Ao longo dos anos a tecnologia tornou-se mais sofisticada e os veículos automatizados de hoje são mais "inteligentes", podem tomar diversas decisões conforme o obstáculo que encontram pela frente.

Os AGV`s são capazes de executar funcionalidades de dirigibilidade num veículo não tripulado, a fim de aumentar a agilidade e segurança das tarefas, reduzindo o numero de funcionários e o desperdício de recurso envolvido.

O presente trabalho tem por objetivo estudar, projetar, desenvolver, construir e testar um sistema de controle de velocidade. No capitulo 2 é feita a descrição geral dos componentes principais que compõem um AGV, com objetivo de enfocar os materiais e técnicas utilizadas neste tipo de projeto. No capitulo 3 está a descrição da arquitetura, dividida em blocos, do protótipo montado com o detalhamento do esquema elétrico da interligação e montagem dos componentes mecânicos e eletrônicos. No capitulo 4 podem ser vistos os resultados dos ensaios em que o protótipo foi submetido, com as técnicas de controle que foram implementadas. O capitulo 5 traz as considerações finais comentando sobre os problemas encontrados o parecer conclusivo e descrevendo algumas sugestões para serem implementadas em trabalhos futuros. O esquema elétrico completo pode ser

visto no Apêndice A e B, no Apêndice C e D estão o fluxograma e *software* respectivamente.

2. VEÍCULOS AUTÔNOMOS E SEUS PRINCIPAIS COMPONENTES

2.1 Motores

Um dos principais elementos que compõe o AGV são os motores. A escolha entre os mais diversos modelos que existem no mercado, sendo algum deles motores de corrente alternada (CA) Síncronos, motores de corrente continua (CC), motores de passo e motores de corrente continua sem escova *Brushless* DC (BLDC), dependente do porte de um ou outro. Motores podem ser alimentados com tensão oriunda de baterias. A seguir mostra-se os motores CC e BLDC.

2.1.1 Motores CC

Na maioria dos motores elétricos CC o rotor é um 'eletroímã' que gira entre os pólos de ímãs permanentes estacionários, para tornar esses eletroímãs mais eficientes o rotor contém um núcleo de ferro, que torna-se fortemente magnetizado quando a corrente flui pela bobina. O rotor girará desde que essa corrente inverta seu sentido de percurso cada vez que seus pólos alcançam os pólos opostos do estator conforme pode ser visto na Figura 1. Nos motores CC o sentido da corrente determina qual será o sentido de sua rotação, podendo ser alterada a sua rotação apenas alterando o sentido da corrente.

O rotor de um motor CC gira com velocidade angular, que é praticamente proporcional à tensão aplicada em suas bobinas. Tais bobinas têm pequena resistência elétrica e conseqüentemente seriam percorridas por intensas correntes elétricas se o rotor permanecesse em repouso, uma vez em movimento, as alterações do fluxo magnético sobre tais bobinas geram uma força contra eletromotriz (FCEM), extraem energia daquela corrente e baixam as tensões elétricas sobre tais bobinas.

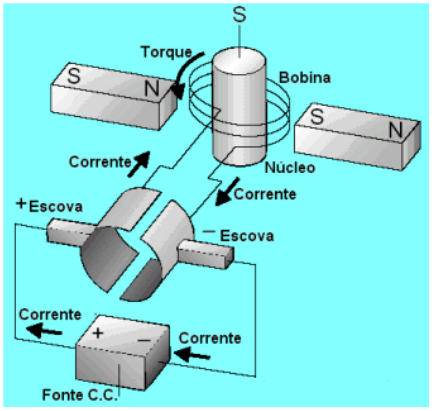


Figura 1 - Motor CC.
Fonte: Feira de Ciências.

O torque resultante se anulará quando essa FCEM se igualar à tensão elétrica aplicada, e então a velocidade angular passa a ser constante.

$$Ece = Ke \times \eta \quad \text{(Eq. 1)}, \qquad Ke = \frac{E}{\eta} \quad \text{(Eq. 2)},$$

$$T = Kt \times Ia \quad \text{(Eq. 3)}, \qquad Kt = \frac{T}{Ia} \quad \text{(Eq. 4)},$$

$$Va = Ece \times IaRa \quad \text{(Eq. 5)}, \qquad Ra = \frac{E}{Ia} \quad \text{(Eq. 6)}.$$

Onde:

Ecf - Tensão contra eletro-motriz;

Ia - Corrente rotor bloqueado;

Ra - Resistência do enrolamento.

Kt - Constante de torque;

E - Tensão rotor bloqueado;

Ke - Constante de alimentação;

Va - Tensão terminal;

T - Torque;

η - Rotação;

Foram analisados e testados dois motores CC que serão referenciados como motor A e motor B a seguir. Na Figura 2 e na Tabela 1 a seguir, sumarizam-se as características do motor A.

Figura 2 - Foto do motor A.

Tabela 1 - Especificações técnicas fornecidas pelo fabricante do motor A.

Especificações técnicas	
Modelo	EdG-M360
Rotação Sem Carga	44rpm
Corrente Sem Carga	0,14 A
Rotação Max. Rend.	33rpm
Corrente Máx. Rend	0,16 A
Torque Máx. Rend.	2,2Kgf/cm
Potência Máx. Rend.	2W
Corrente de Partida	0,41 A
Torque de Partida	8,6Kgf/cm
Tensão de Operação	12V ~ 25V
Tensão Nominal	12V

Este motor foi caracterizado a partir de três ensaios: a vazio como gerador; motor com rotor bloqueado; motor a vazio. Abaixo seguem os dados obtidos na Tabela 2 abaixo.

Tabela 2 - Tabela dos valores do teste de caracterização do motor A.

Ensaio a Vazio - Gerador	**η –Rotação[rpm]**	**Tensão (V)**	
	31,5	7	
	50	11,08	
Ensaio Rotor Bloqueado	**E – Tensão (V)**	**Ia - Corrente(A)**	
	12,32	0,575	
Ensaio Motor a vazio	**Rotação (RPM)**	**Tensão (V)**	**I Carga (A)**
	47	11,61	0,046
		12,45	0,058

Atribuindo alguns valores constantes equações descritas anteriormente Eq.1 à Eq.6 pode-se analisar os gráficos abaixo, representados nos Gráficos 1 e 2, rotação e torque e potência e torque respectivamente.

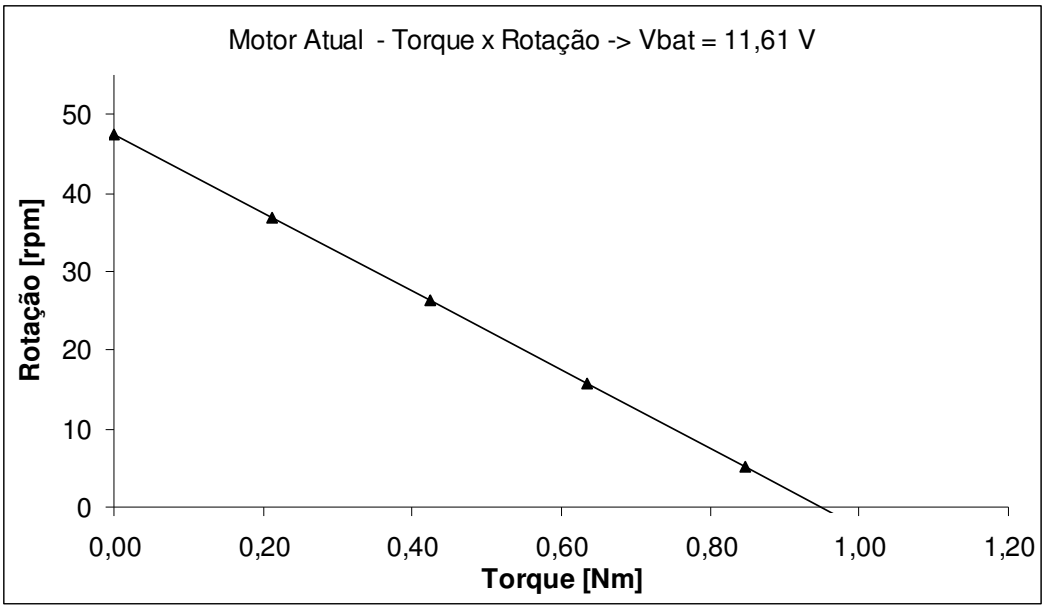


Gráfico 1 - Análise da relação entre rotação por torque do motor A.

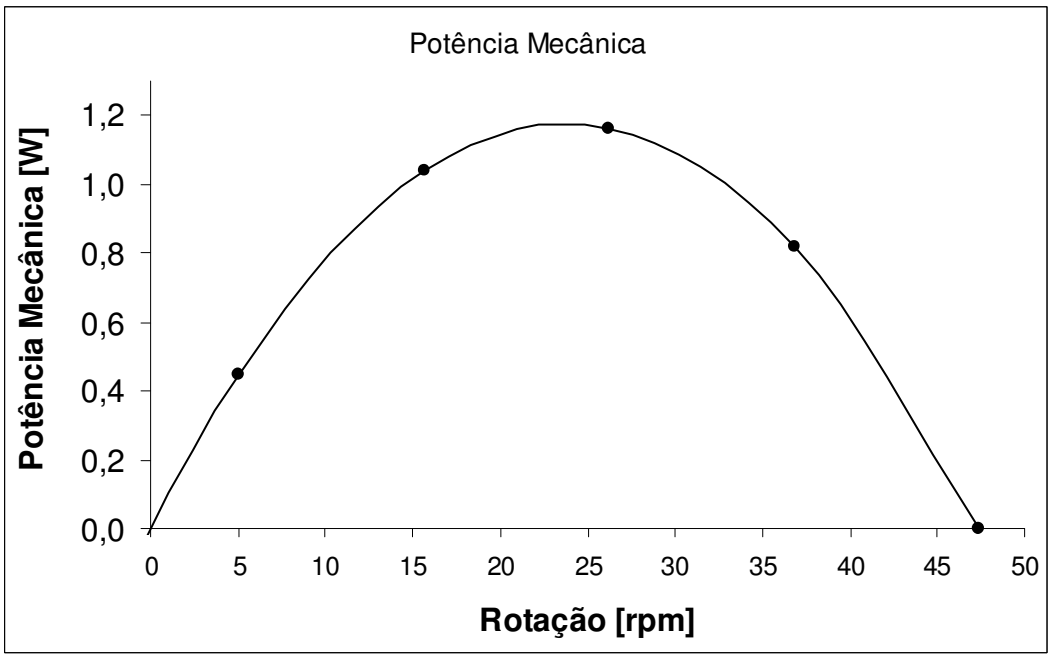


Gráfico 2 - Análise da potência mecânica pela rotação do motor A.

Na Figura 3 e na Tabela 3 a seguir sumarizam-se as características do motor B, com os dados fornecidos pelo fabricante sendo o motor B testado.

Figura 3 - Foto do motor B.

Tabela 3 - Especificações técnicas fornecidas pelo fabricante do motor B.

Especificações Técnicas	
Modelo	QJT-51JS-A
Rotação Sem Carga	600rpm
Potência Máx. Rend.	7W
Tensão de Operação	3V ~ 36V
Tensão Nominal	12V

A caracterização do motor B seguiu o padrão de ensaios conforme mencionados ao motor A: a vazio como gerador; motor com rotor bloqueado; motor a vazio. Abaixo seguem os dados obtidos na Tabela 4.

Tabela 4 - Tabela dos valores do teste de caracterização do motor B.

Ensaio a Vazio - Gerador	**η - Rotação[rpm]**	**Tensão (V)**	
	500	4,5	
	1000	8,95	
Ensaio Rotor Bloqueado	**E - Tensão (V)**	**Ia - Corrente(A)**	
	7,4	7,35	
Ensaio Motor a vazio	**Rotação (RPM)**	**Tensão (v)**	**I Carga (A)**
	1130	11,64	0,65

Com análise das equações Eq.1 à Eq.6 descritas acima e atribuindo valores constantes obtêm-se os gráficos das relações entre rotação e torque e potência e torque do motor B representados nos Gráficos 3 e 4 respectivamente.

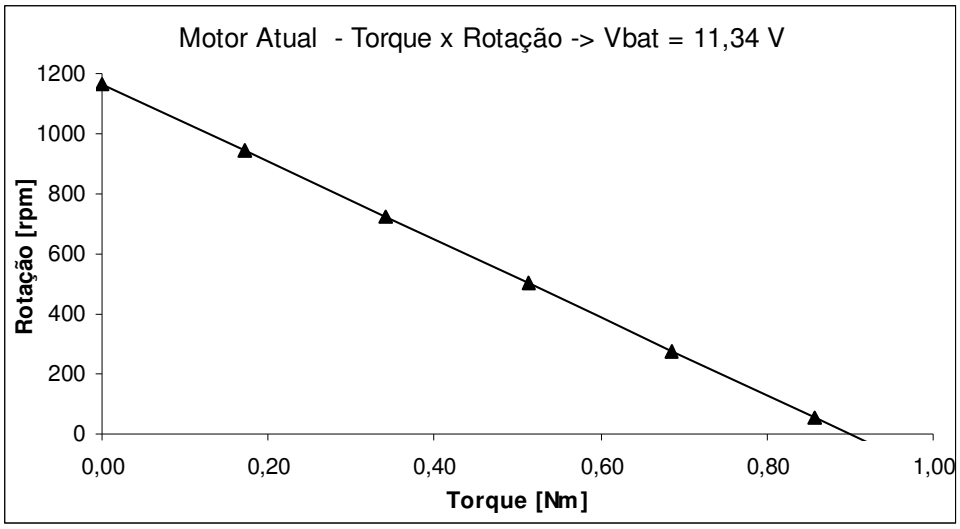


Gráfico 3 - Análise da relação entre rotação por torque do motor B.

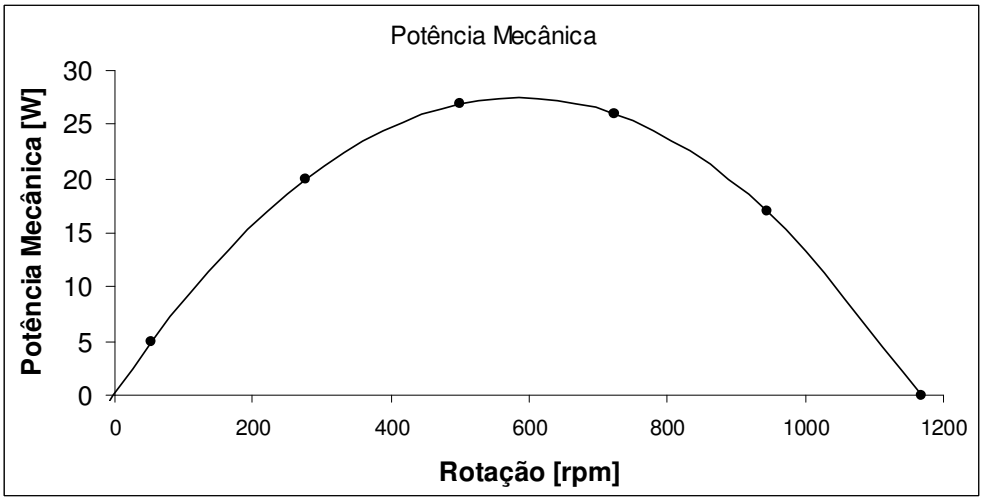


Gráfico 4 - Análise da potência mecânica pela rotação do motor B.

O motor A foi escolhido para ser utilizado neste trabalho por apresentar um torque elevado em baixas rotações. Essa característica possibilita aplicar um nível de tensão de 4V em seus terminais resultando em um torque suficiente para o deslocamento do protótipo atendendo a necessidade deste trabalho. Diferentemente do motor B o motor, A por ter uma redução menor, resulta em uma rotação alta necessitando de uma tensão mais elevada para simplesmente vencer a inércia mecânica do próprio motor.

2.1.2 Motores BLDC

Os motores de corrente contínua BLDC oferecem diversas vantagens sobre os motores de corrente contínua com escovas. Dentre as quais destacam-se confiabilidade mais elevada, o ruído reduzido e a vida útil mais longa (devido a ausência da escova).

A desvantagem principal do motor sem escovas é o custo mais elevado e estes motores requerem dispositivos de acionamento conhecidos como *drivers.*

Os motores BLDC são considerados mais eficientes em "baixa-carga" do que os motores de corrente contínua escovados. Isso significa que, para a mesma potência de entrada os motores BLDC converterão mais energia elétrica em energia mecânica do que um motor de corrente contínua escovado.

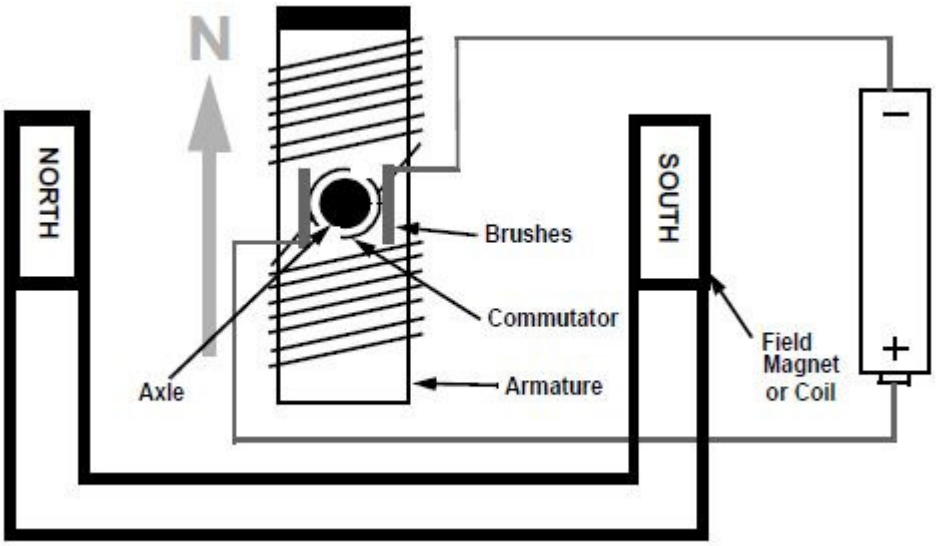


Figura 4 - Motor BLDC.
Fonte - Microchip AN905.

Não foi testado nenhum motor BLDC, pois não foi encontrado nenhum que atendesse à especificação de torque compatível com os motores CC testados.

2.2 Acionamento

Para o acionamento de motores CC é necessário utilizar um circuito acionador conhecido como *driver*, os *drivers* podem estar na configuração de Ponte H.

2.2.1 Ponte H

Quando ligado um motor CC observa-se que ele gira numa velocidade constante e em uma única direção, para alterar o sentido da rotação do motor deve-se ligar os terminais do motor de forma invertida. A utilização de ponte H para esta inversão pode ser confeccionada utilizando chaves simples ou semicondutores, bastando apenas entender o seu funcionamento. Uma ponte H básica é composta por 4 chaves mecânicas ou eletrônicas posicionadas formando a letra "H", sendo que cada uma localiza-se num extremo e o motor é posicionado no meio.

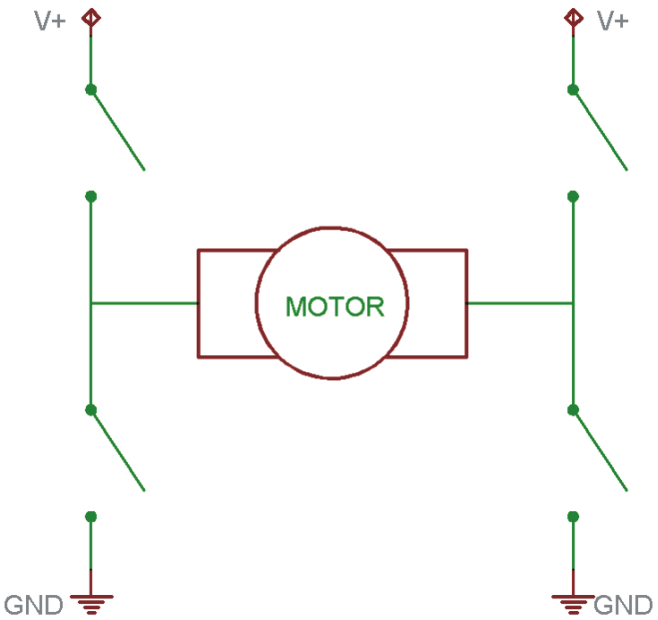


Figura 5 - Representação da Ponte H.
Fonte: Wikipédia.

Para que o motor funcione basta acionar um par de chaves diagonalmente opostas, o que faz com que a corrente flua do pólo positivo para o negativo, atravessando o motor e fazendo-o girar. Para inverter a rotação essas chaves dever ser desligadas acionando o outro par de chaves, o que faz com que a corrente siga na direção oposta e, conseqüentemente, o sentido da rotação do motor será alterada. Esse tipo de acionamento (ponte H) pode ser encontrado em CIs monolíticos. Estes *drivers* podem ser configurados como ponte completa ou meia ponte para o seu funcionamento.

Neste projeto foi adotado o sentido apenas em uma direção, com isto o circuito de acionamento pode ser de meia ponte.

2.3 *Driver* de acionamento dos motores

Para o acionamento dos motores foi utilizado o circuito conhecido como *push-pull driver* o mesmo pode ser encontrado no formato de CI. O CI utilizado foi o L293B do fabricante *SGS-Thomson,* a escolha deste foi em função da corrente de acionamento dos motores que é inferior a 0.5A.

Podemos ver na Figura 6 a pinagem do componente, que contém quatro saídas *push-pull* (out1 à out4), dois pinos de habilitação (pinos 1 e 9), e os pinos de acionamento (2, 7, 10 e 15) individual para cada saída *push-pull*, a alimentação do CI é de +5V (pino 16) e a alimentação dos motores é de +12V.

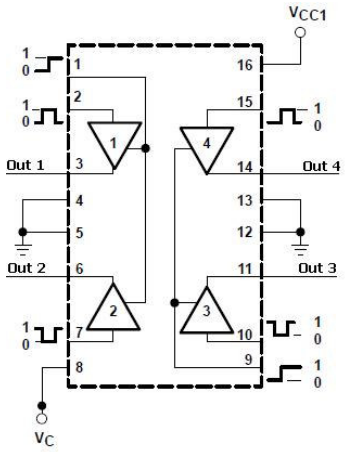


Figura 6 - Pinagem do L293B.
Fonte: *Datasheet* L293B.

2.4 Sistema de Controle

Nos motores CC a velocidade de rotação pode ser controlada através da variação da corrente de campo ou da variação da corrente de armadura.

Consegue-se controlar a corrente de armadura variando a tensão aplicada nos terminais do enrolamento de campo, alterando a amplitude da tensão DC ou através da modificação do valor médio da tensão.

2.4.1 Acionamento PWM

A técnica *Pulse Width Modulation* (PWM), foi desenvolvida para aplicações em telecomunicações e, posteriormente, passou a ser utilizada na eletrônica de potência como, por exemplo, no controle de tensão sobre o enrolamento de motores.

O acionamento pode ser feito através da variação da largura de pulso (PWM), a partir da saída de atuação (modulação) de um controlador Proporcional – Integral – Derivativo (PID). A seguir discute-se o acionamento PWM e os controladores PID.

Para se entender como funciona esta tecnologia de controle de potência, vamos iniciar com o seguinte circuito mostrado na Figura 7, que ilustra o principio básico de funcionamento através de um interruptor, que tem sua ação determinada por um circuito de controle.

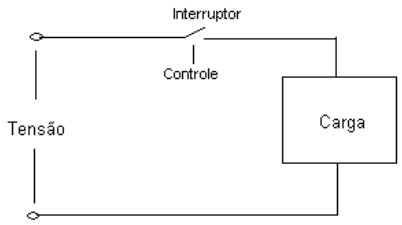


Figura 7 - Circuito ilustrativo para PWM.

No circuito ilustrativo mostrado na Figura 7, quando o interruptor está aberto não há passagem de corrente para a carga, por consequência a potência é zero. Quando o interruptor é fechado a carga recebe a tensão total e por conseqüência a potência é máxima.

Esse princípio de abertura e fechamento de interruptor é justamente o que acontece no PWM. Para controlar a potência na carga basta fazer com que o interruptor fique alternando entre os ciclos ativos (*duty cycles*). Para obter metade da potência aplicada na carga, basta fazer com que os tempos t_1, ligado, e t_2, desligado, sejam iguais.

A frequência da onda é definida com o número de ciclos por unidade de tempo e o tempo como período que será a soma do tempo ligado e o tempo desligado.

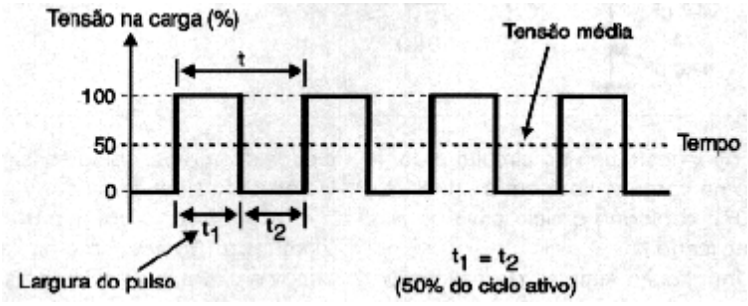


Figura 8 - Forma de onda do PWM com 50%.
Fonte: [15].

O *duty cycle* define a duração do sinal ligado, conforme mostrado na Figura 9 pode ver diferentes formas de ondas, o primeiro sinal com 10% de *duty cycle*, ou seja, durante 10% do período o sinal ficará ligado e 90% desligado, sendo a tensão média na carga de 10% do valor nominal. O segundo e terceiro sinal, agora com 40% e 90% de *duty cycle,* terão a tensão média diferente, de acordo com a largura do pulso.

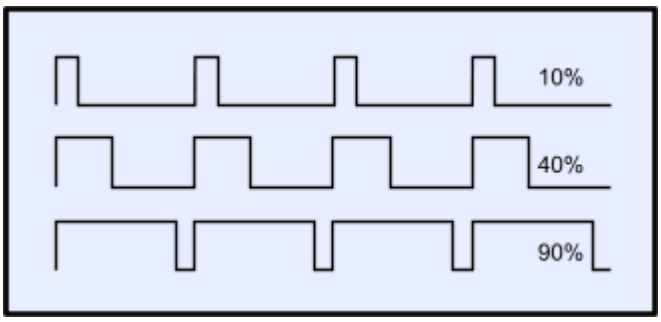


Figura 9 - Representação de formas de onda com *duty cycles* diferentes.
Fonte: [15].

2.4.2 Controladores P, I e D

Um sistema de controle é, basicamente, um sistema de entradas e saídas conforme Figura 10.

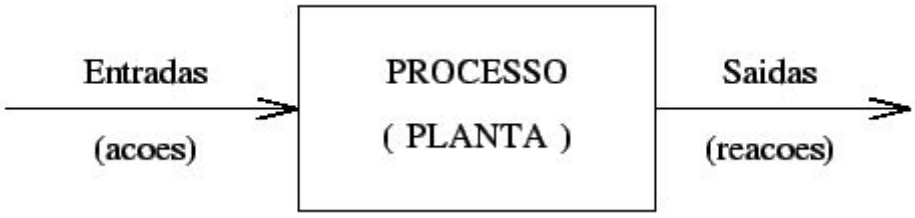


Figura 10 - Sistema de Controle.

O sistema a ser controlado é, em geral, chamado de *processo* ou *planta*. O processo é um sistema dinâmico, ou seja, seu comportamento é descrito matematicamente por um conjunto de equações diferenciais. Como exemplos de sistemas dinâmicos temos entre outros: sistemas elétricos, mecânicos, químicos. A entrada do processo $u(t)$ é chamada de variável de controle ou variável manipulada (MV) e a saída do processo é chamada de variável controlada ou variável de processo (PV). A filosofia básica de um sistema de controle consiste em aplicar sinais adequados na entrada do processo, com o intuito de fazer com que o sinal de saída satisfaça as especificações e/ou apresente um comportamento particular. Um problema de controle consiste, então, em determinar os sinais adequados a serem aplicados a partir da saída desejada e do conhecimento do processo.

No controle em malha fechada, informações sobre como a saída de controle está evoluindo, são utilizadas para determinar o sinal de controle, que deve ser aplicado ao processo em um instante específico. Isto é feito a partir de uma realimentação, *feedback* da saída para a entrada. Em geral, a fim de tornar o sistema mais preciso e de fazer com que ele reaja a perturbações externas, o sinal de saída é comparado com um sinal de referência (*set-point*) e o desvio (erro) entre estes dois sinais é utilizado para determinar o sinal de controle, que deve efetivamente ser aplicado ao processo. Assim, o sinal de controle é determinado de forma a corrigir este desvio entre a saída e o sinal de referência. O dispositivo que utiliza o sinal de erro para determinar ou calcular o sinal de controle a ser aplicado à planta é chamado de controlador ou compensador. O diagrama básico de um sistema de controle em malha-fechada é mostrado na Figura 11.

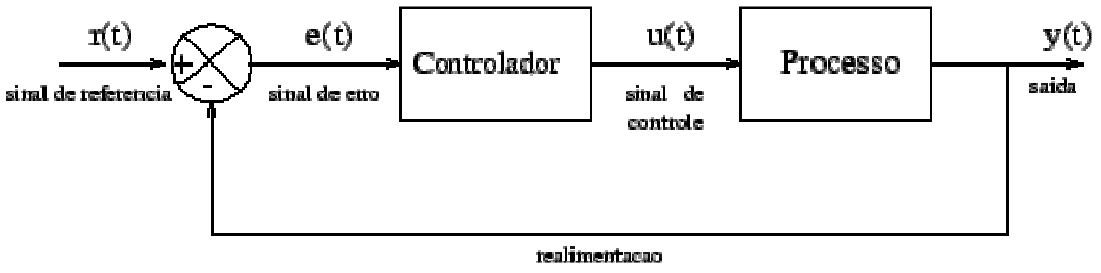


Figura 11 - Controle em malha fechada.

O controlador pode ser entendido como um dispositivo que realiza determinadas operações matemáticas sobre o sinal de erro $e(t)$, a fim de produzir um sinal $u(t)$ a ser aplicado à planta, com o intuito de satisfazer um determinado

objetivo. Estas operações matemáticas constituem o que chamamos de ações de controle e podemos identificar 3 ações básicas de controle:

- Ação Proporcional (P);
- Ação Integral (I);
- Ação Derivativa (D).

Ação Proporcional (P)

Muitas vezes processos simples podem ser controlados satisfatoriamente apenas com a ação proporcional. Neste caso a ação integral e derivativa é simplesmente desligado sendo o objetivo de levar para zero o sinal de erro $e(t)$.

A saída do controlador proporcional tem:

$$u(t) = Kp.e(t) \qquad \text{(Eq. 7)}$$

Onde Kp é uma constante chamada de ganho proporcional, normalmente adimensional. O ganho define o quanto a variável de controle deve variar em correspondência a uma variação unitária do sinal de erro.

Ação Integral (I)

A idéia básica é definir um controlador tal que sua saída permaneça constante quando o sinal de erro é nulo. Uma maneira de conseguir esta característica é definindo a saída do controlador como sendo proporcional à integral do sinal de erro ao longo do tempo, isto é:

$$u(t) = ki\int_0^t e(t)dt \qquad \text{(Eq. 8)}$$

Onde ki é uma constante chamada de ganho integral e tem dimensão de tempo.

Ação Derivativa (D)

A ação de controle derivativa tem um caráter antecipatório, sendo sua função reagir antecipadamente ao comportamento futuro do sinal de erro com base na sua taxa de variação.

A ação derivativa ideal pode ser expressa por:

$$u(t) = kd\frac{de(t)}{dt} \qquad \text{(Eq. 9)}$$

em que Kd é o ganho derivativo e tem dimensão de tempo. Dessa maneira, o avanço produzido pelo termo derivativo pode compensar o atraso introduzido por praticamente todas as malhas de controle.

Controle PID

A combinação entre as três formas de controle, ação proporcional, ação integral e ação derivativa, formam o tradicional controlador PID expressa por:

$$Gc = kp.e(t) + ki\int_0^t e(t) + kd\frac{e(t)}{dt} \qquad \text{(Eq. 10)}$$

A função de transferência do controlador PID é dada por:

$$Gc(s) = kp + \frac{ki}{s} + kds \qquad \text{(Eq. 11)}$$

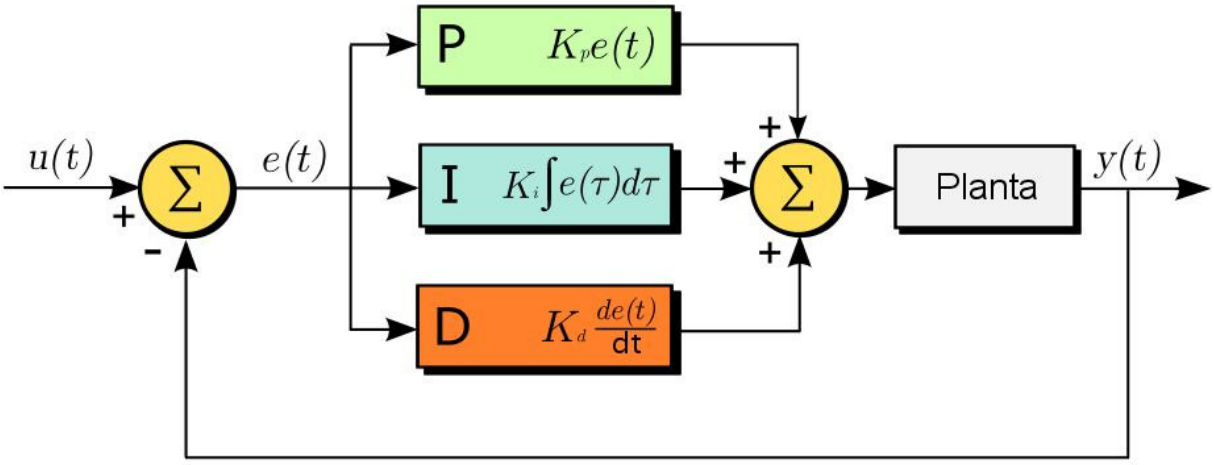


Figura 12 - Controlador PID
Fonte: Wikipédia.

2.5 Codificadores (*Encoders*)

Há diversos tipos de sensores para obter a velocidade dos AGV`s, entre eles estão os *encoders*.

Os *encoders* são transdutores de movimento capazes de converter movimentos lineares ou angulares em pulsos elétricos. Estes que podem ser

transformadas em informações binárias e operadas através de *software* que converta as informações em velocidade.

Os *encoders* do tipo óptico possuem internamente um ou mais discos (máscaras) perfurados, o que permite, ou não, a passagem de um feixe de luz infravermelha gerado por um emissor que se encontra de um dos lados do disco e captado por um receptor que se encontra do outro lado do disco, esse, com o apoio de um circuito eletrônico gera um pulso.

Há diferentes tipos de *encoders*, os mais usuais são: simples, incremental e absoluto.

2.5.1 *Encoder* Simples

O encoder simples é composto por apenas um emissor e um receptor infravermelho e a cada interrupção da luz, devido a rotação do disco, um pulso é enviado.

Da freqüência dos pulsos determina-se a velocidade do eixo e a quantidade de pulsos determina a posição a partir de um referencial.

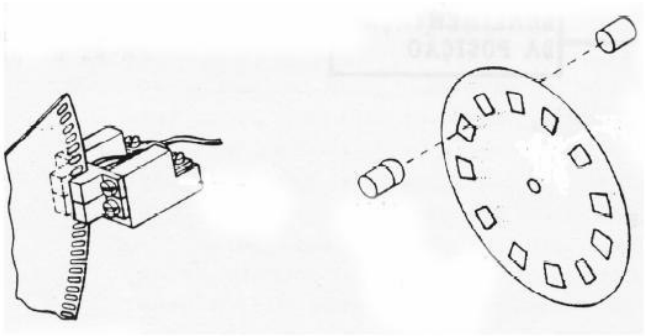


Figura 13 - *Encoder* Simples.
Fonte: Utilização de sensores e transdutores ópticos.

2.5.2 *Encoder* Incremental

Neste tipo de encoder a posição é demarcada através de pulsos transmitidos e acumulados ao longo do tempo.

Esses pulsos, quadrados, são transmitidos pelo encoder através de dois canais, 1 e 2, defasados de 90°. Para ler apenas a posição pode-se utilizar um dos canais, 1 ou 2, indistintamente. Se for necessário saber o sentido do movimento,

deve-se utilizar os dois canais simultaneamente. Em função da defasagem de 90° entre os canais 1 e 2, pode-se saber o sentido de rotação ou deslocamento do encoder. Caso o canal 1 esteja 90° adiantado em relação ao canal 2 o sentido será horário e se o canal 1 estiver atrasado 90° em relação ao canal 2 o sentido será anti-horário. Existe outro canal, de sincronismo, também chamado de "zero" do *encoder*. Ele fornece uma posição de referência, gerando um pulso quadrado a cada revolução do *encoder*.

A resolução do *encoder* incremental é dada por pulsos por revolução (PPR), isto é, o *encoder* gera uma certa quantidade de pulsos elétricos por uma revolução dele próprio (no caso de um *encoder* rotativo).

Para determinar a resolução basta dividir o número de pulsos por 360°.

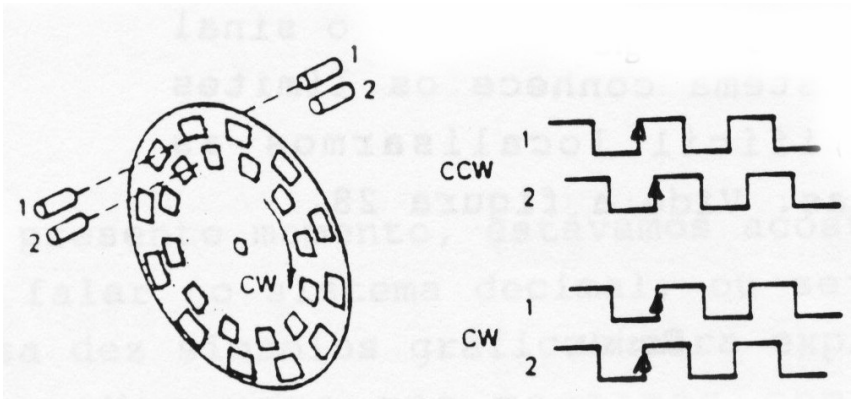


Figura 14 - *Encoder* incremental.
Fonte: Utilização de sensores e transdutores ópticos.

2.6 Unidade de processamento e controle

A utilização de um sistema central para o controle é necessária nos projetos de *AGV`s*, podemos dizer que é essencial. Em geral o sistema central é implementado a partir de um micro-processador/micro-controlador digital. Neste projeto foi utilizado um micro-controlador da linha PIC18F.

As funções desse micro-controlador e suas particularidades serão abordadas no Capítulo 3, onde a implementação de *hardware* e *software* será mostrada.

3. ARQUITETURA E PARTES DO PROTÓTIPO

Os materiais e métodos utilizados pra o desenvolvimento deste projeto serão descritos neste capitulo, que contém o detalhamento da montagem das peças mecânicas, componentes eletrônicos, *hardware*, *software* e seu fluxograma.

3.1 Descrição Geral do Sistema

O funcionamento simplificado do projeto se dá em 3 etapas:

i) Leitura dos valores: onde são lidos os valores de velocidade setado pelo usuário através das chaves (*dips*) e a velocidade de cada roda através dos *encoders ópticos*. Calculado o erro entre essas velocidades.

ii) Controle da velocidade: consiste em aplicar o erro em um controlador PI que será implementado via *software* e converter em porcentagem de PWM variável para cada roda.

iii) Acionamento: acionam os motores através de um circuito integrado de potência conhecido como *push-pull driver,* responsável pela ligação entre micro-controlador e os motores.

A ligação das três etapas efetua o fechamento da malha de controle. O diagrama de blocos simplificado pode ser visto na Figura 15 a descrição detalhada de cada item será abordada nas próximas seções.

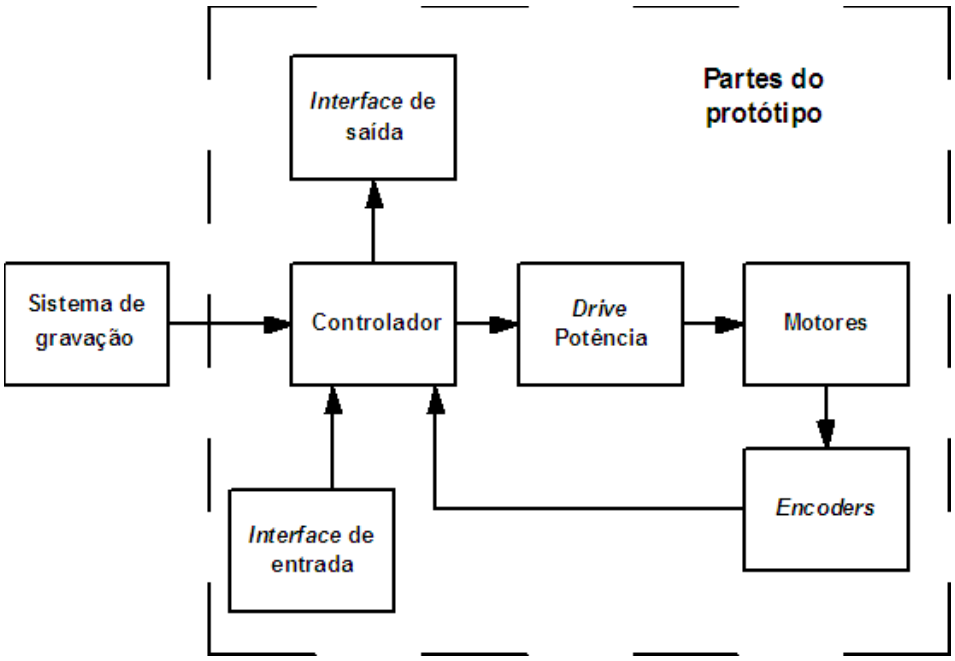


Figura 15 - Diagrama de blocos simplificado.

Na figura 16 abaixo pode ser visto o protótipo montado como os principais elementos destacados e que serão comentados a seguir.

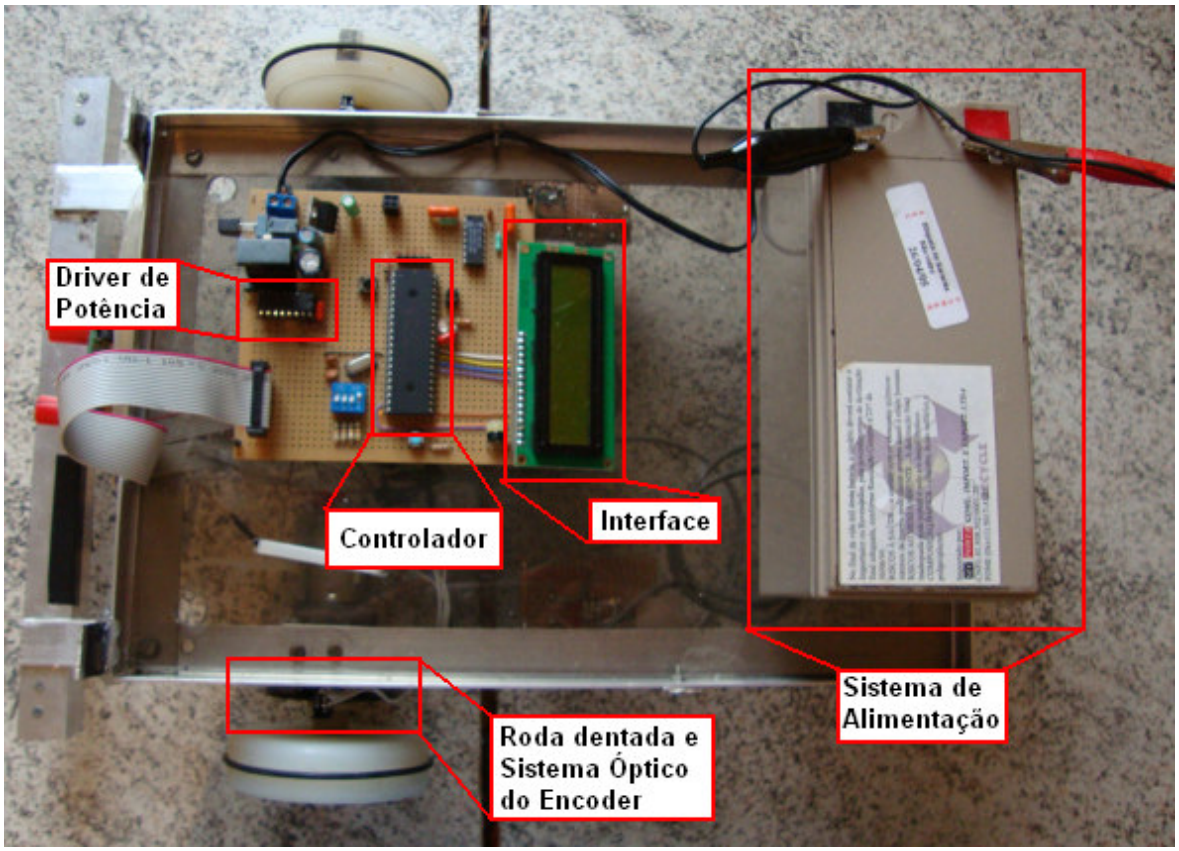


Figura 16 - Protótipo montado com os componentes destacados.

O sistema de alimentação por bateria de $12V_{dc}$ mostrado na Figura 16 foi substituído por uma fonte de alimentação fixa, reduzindo o peso do protótipo o que possibilitou uma performance melhor na velocidade final.

3.2 O Micro-controlador

Para controlar as informações, ou sinais, deste projeto foi utilizado o micro-controlador PIC18F4620, do fabricante *Microchip Technology Inc* encapsulado em um CI de 40 pinos. É o componente principal deste projeto, pois é ele quem gerencia todas as informações dos *hardwares* periféricos.

O micro controlador é da família PIC18MCU, tem capacidade de execução de instruções de 4 ciclos de *clock*, ou seja 20Mhz, resultando no tempo de 0.2µs para cada instrução, além de 64kB de memória de programa, 4 *Timers* interno, 2 módulos *Capture*/Compare/PWM (CCP) para implementação de PWM e quatro *bytes* de pinos de I/O bidirecional, configurados como digital e alguns podendo ser configurados como analógicos. A pinagem do micro-controlador pode ser vista na Figura 17 abaixo.

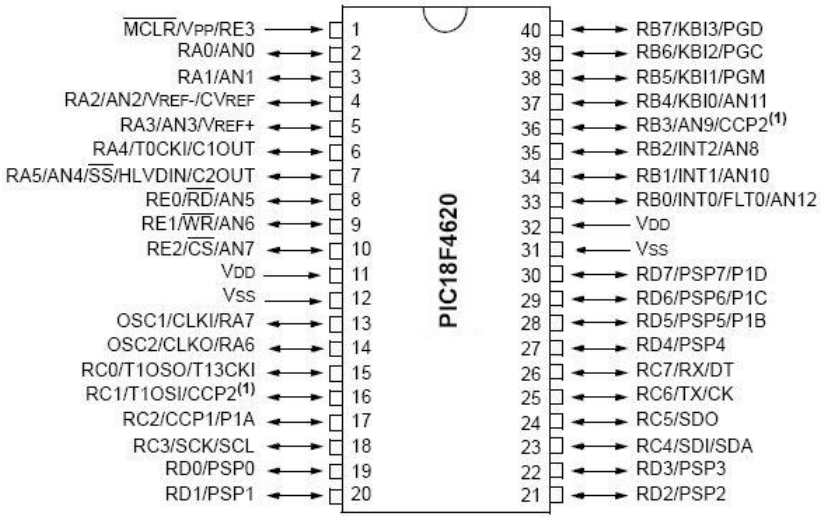


Figura 17 - Pinagem micro-controlador PIC18F4620
Fonte - *Datasheet* PIC18F4620.

3.3 Pinos de I/O

Na Figura 17 pode ser vista a nomenclatura geral dos pinos do PIC, as portas de I/O são organizadas em cinco portas, quatro de oito *bits* e uma de quatro *bits*, alguns destes pinos são utilizados como pinos dedicados para outras funções.

No projeto foram utilizadas as portas de I/O da seguinte forma:

Gravação do *software*: dois pinos da porta B (RB6 e RB7) e um da porta E (RE3) configurados como entrada.

Comunicação com o *display,* sete pinos da porta D (RD0 à RD2 e RD4 à RD7) configurados como saída.

Saída do PWM: dois pinos da porta C (CCP1 e CCP2) configurados como pinos dedicados CCP.

Entrada do oscilador: dois pinos da porta A (OSC1 e OSC2) configurados como pinos dedicados.

Interrupções externas: dois pinos da porta B (RB4 e RB5) configurados como entrada de interrupção externa chamado de "RB *Port Change Interrupt Enable bit*", que tem como característica gerar um pedido de interrupção a cada transição de borda.

3.4 Sistema de Gravação do Firmware

Para a gravação do *firmware* no PIC tendo a possibilidade de ser *in-circuit* através do kit de gravador *MicroICD* do fabricante *MicroGênios.*

A facilidade de ser *in-circuit* possibilita a gravação do micro-controlador sem a necessidade de retirar do circuito, o Kit de gravador *MicroICD* pode ser visto na Figura 18.

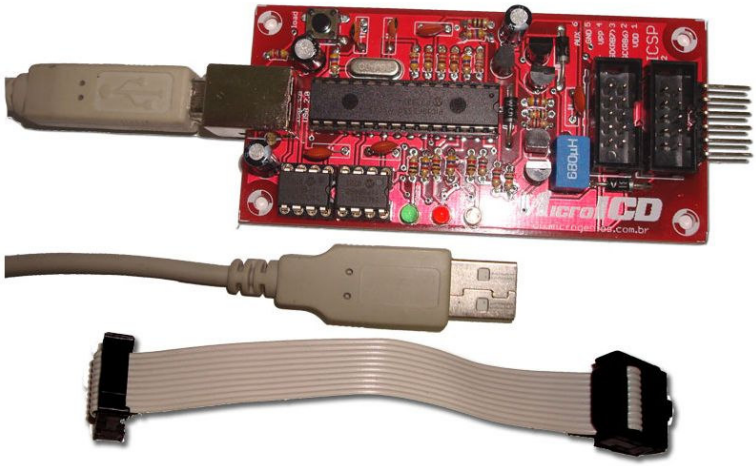


Figura 18 - Gravador *MicroICD*.
Fonte: Manual MicroICD.

O *firmware* desenvolvido em linguagem de alto nível C, foi descrito e compilado pelo compilador *MicroC* da *mikroElektronika,* que será detalhado no capitulo 3.4.1. Para a gravação foi utilizado o *software* que acompanha o kit de gravador que será detalhado no capitulo 3.4.2.

3.4.1 O Compilador *MicroC*

O compilador *MicroC* consiste em um ambiente de desenvolvimento para toda a linha de micro-controladores PIC (series PIC12, PIC14, PIC16 e PIC18), com funções de *debugger,* o qual é responsável pela compilação e geração do *.hex*, que será enviado ao micro-controlador.

3.4.2 *Software* de Gravação

O programa para gravar o PIC é o PICKit2 que acompanha o kit de gravador de PIC *MicroICD*. Uma das funções é a identificação automática do modelo do PIC que está sendo gravado, conforme pode ser visto circulado as mensagens abaixo informam que o PIC está pronto para ser gravado, conforme Figura 19 da interface de gravação do PICkit2.

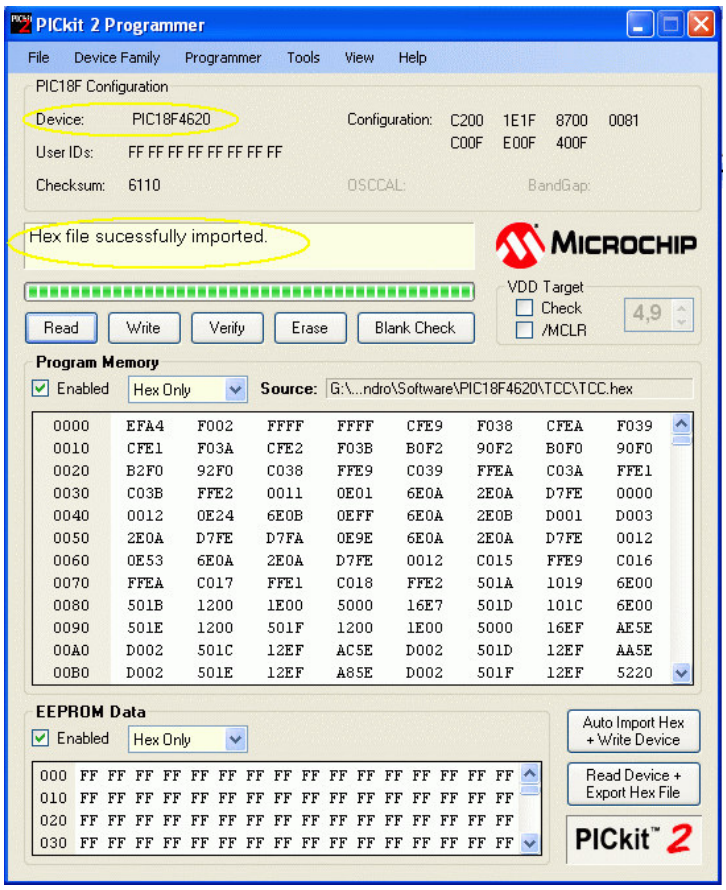


Figura 19 - Interface de gravação do PIC PICkit2.

3.5 Interface com o usuário

Como interface de entrada foi utilizado uma *dip switch* para a escolha/configuração do *set point* de velocidade, esta implementação possibilita a escolha do *set point* sem ter que atualizar via *software* a cada necessidade de alteração. A configuração das *dips* seguem a seqüência mostrada na Tabela 5 abaixo.

Tabela 5 - Interface de entrada, configuração do *set point* de velocidade.

DIP		*Set point* de velocidade (m/min)
CH1	CH2	
OFF	OFF	4,3
ON	OFF	5
OFF	ON	8
ON	ON	10

O módulo LCD é uma interface de saída utilizada para visualização das informações, é o WH1602A do fabricante *Winstar*, é do tipo caráter 2x16 (duas linhas e 16 colunas), tem um controlador LCD modelo KS0066. Na Tabela 6 está a descrição dos pinos para a interface de dados.

Tabela 6 - Descrição dos pinos do display LCD.

Fonte: *Datasheet* WH1602A.

Pino	Nome	Função
1	Vss	Terra
2	Vdd	+V5
3	Vo	Contraste do LCD
4	Rs	*Register Select*
5	R/W	*Read/Write*
6	E	*Enable*
7	D0	Bit 0
8	D1	Bit 1
9	D2	Bit 2
10	D3	Bit 3
11	D4	Bit 4
12	D5	Bit 5
13	D6	Bit 6
14	D7	Bit 7
15	A	Anodo do *Back-light*
16	K	Catodo do *Back-light*

A interligação entre o *display* e o micro-controlador foi no modo de operação de 4 *bits*, o qual permite a redução nos pinos utilizados do micro-controlador, utilizando apenas quatro pinos de saída para os dados utilizando no *display* os *bits* 4 a 7. O esquema elétrico da ligação entre o *display* e o micro-controlador pode ser visto na Figura 20.

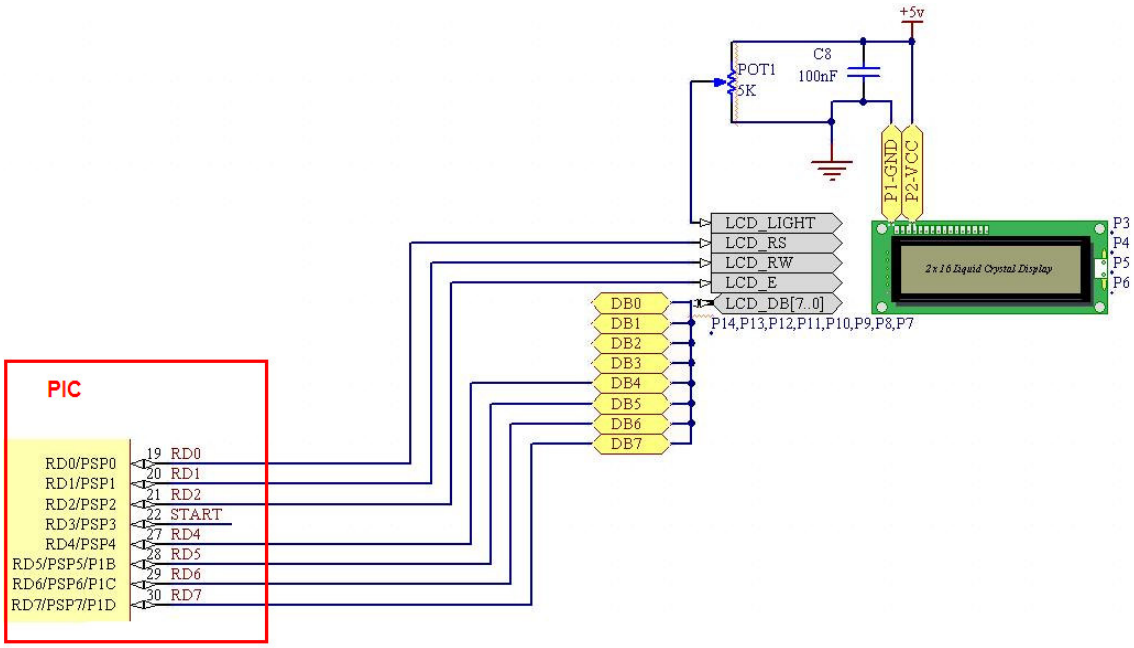


Figura 20 - Esquema elétrico do *display* e o micro-controlador.

3.6 Conjunto mecânico e *encoders*

3.6.1 Motores

Após os testes a escolha foi do motor A com caixa de redução integrada, a escolha deste motor foi pela necessidade de ter torque em baixa rotação. O motor B não apresentou um torque razoável

Pode ser visto na Figura 21 a ligação dos componentes para o acionamento dos motores. No *insert* vermelho observa-se os pinos do micro-controlador, a ligação do *driver* (L293B) com os motores.

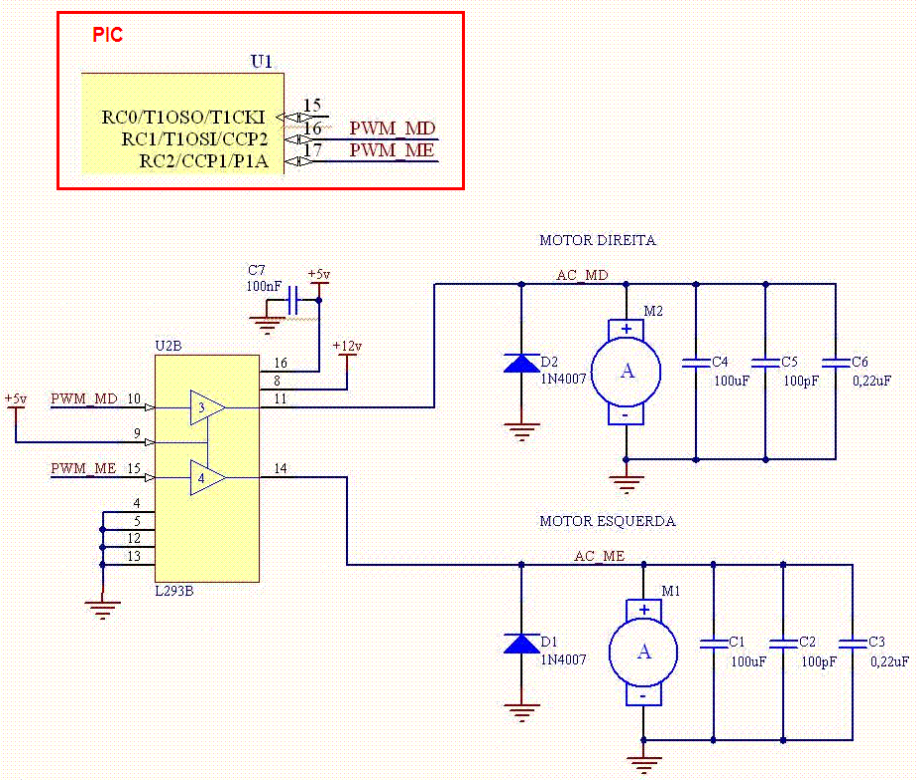


Figura 21 - Esquema elétrico do acionamento dos motores.

Os capacitores C1 a C6 foram aplicados como filtro para minimizar o ruído causado pelos motores, será detalhado a seguir.

3.6.2 Encoder

O sistema de *encoder* é separado em duas partes, o disco perfurado e o sistema óptico. O disco perfurado é responsável por bloquear a passagem da luz, a resolução do *encoder* é dada pela quantidade de fendas existentes neste disco. O disco utilizado é de 50 resoluções por volta, isso quer dizer que a resolução deste encoder será 50 de pulsos em uma volta. Quanto maior for essa relação maior a precisão obtida. Nesse caso temos a seguinte relação angular de:

$$pv = \frac{2\Pi}{n} \qquad \text{(Eq. 12)}$$

Onde:

pv - pulso por volta;

n - quantidade de pulsos.

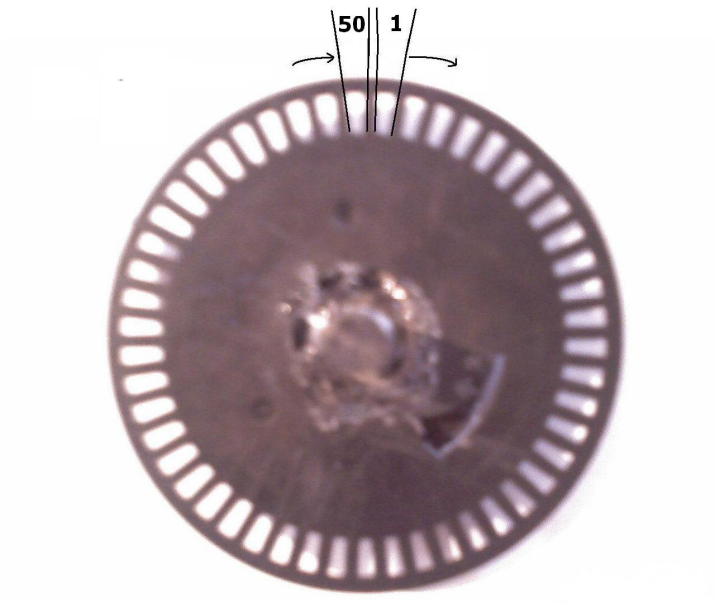


Figura 22 - Imagem do disco perfurado utilizado.

O sistema óptico utilizado foi o EE-SG3 do fabricante *Omron*, por ser encapsulado em um único componente, conforme pode ser visto na Figura 23, o *led* chamado de emissor e um foto transistor conhecido como receptor.

Atendendo as especificações do fabricante o seguinte circuito foi projetado.

Temos:

$$R18 = \frac{V - ve}{ie} \qquad \text{(Eq. 13)}$$

$$R19 = \frac{V - vr}{ir} \qquad \text{(Eq. 14)}$$

Onde:

V - tensão de alimentação; *ie* - corrente do emissor;

ir - corrente do receptor; *ve* - tensão do emissor;

vr - tensão do receptor.

$$R18 = \frac{5v - 4v}{10mA} = 100\Omega \qquad \text{(Eq. 15)}$$

e

$$R19 = \frac{5v - 0{,}2v}{4{,}8mA} = 1k\Omega \qquad \text{(Eq. 16)}$$

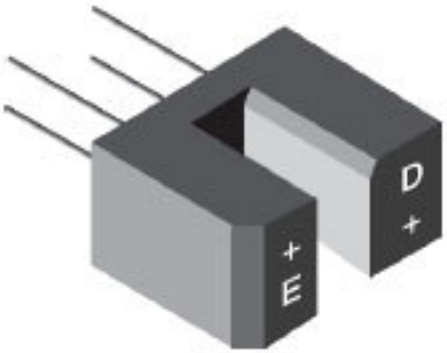


Figura 23 - Foto do sistema óptico encapsulado.
Fonte: *Datasheet* do componente.

Um dos problemas encontrados com o sinal do *encoder* foram os ruídos ocasionados pelos motores, os motores utilizados contêm escovas que em seu funcionamento ocasionam interferência no micro-controlador que gerava acionamentos indesejados e, pode ser visto o sinal do *encoder* sofrendo interferência com o ruído na Figura 24 abaixo.

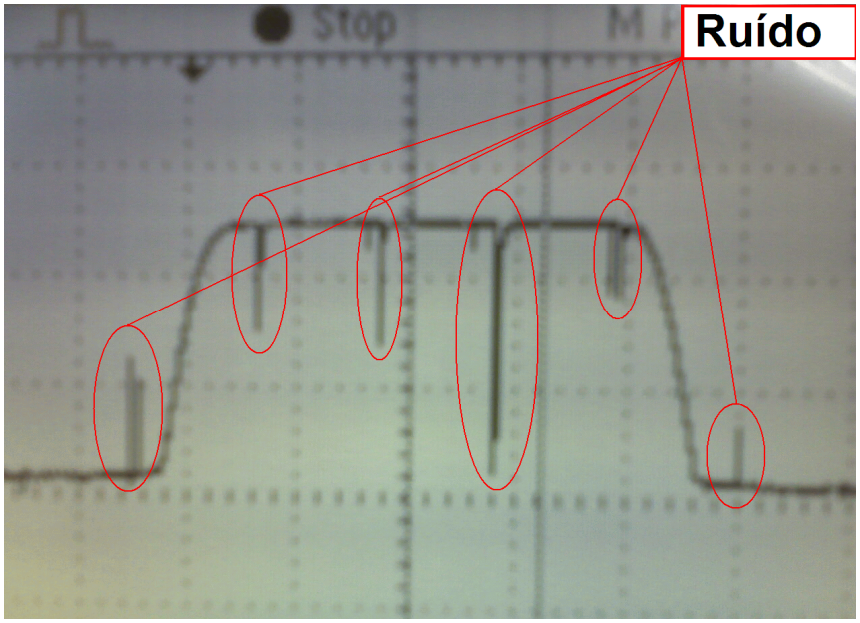


Figura 24 - Ruído gerado pelo motor.

Para a solução deste problema foram adicionados três capacitores em paralelo nos terminais de cada motor, com valores de C1 a C3 de 100pF, 100nF e 0,22uF solucionando o problema dos acionamentos indesejados, conforme Figura 25.

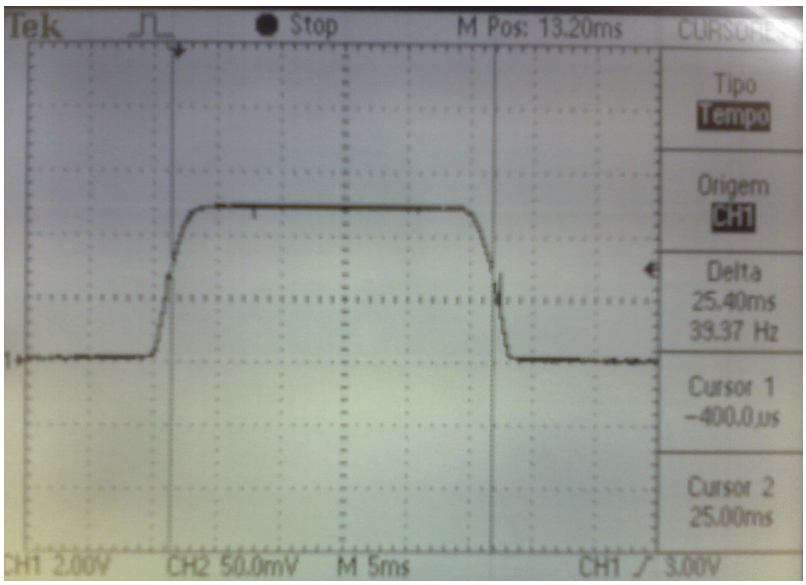


Figura 25 - Acionamento dos motores sem ruído.

Na Figura 25 as bordas de subida e de descida não são abruptas, tendo um tempo de subida e de descida de aproximadamente 2,5ms, para a solução desse problema foi utilizado um circuito com inversoras com *Schimitt trigger*, o CI utilizado foi o 74HCT14 do fabricante *PHILIPS*.

Para o condicionamento do sinal foram utilizadas duas portas inversoras para cada sinal, com isso foi mantido o sinal no mesmo nível lógico da entrada não utilizando a função inversora no sinal na saída.

Além dos capacitores adicionados nos terminais dos motores foi adicionado um filtro RC Passa-Baixa na placa de processamento este filtro esta foi adicionado antes das portas inversoras na analise da Figura 24 notamos que o ruído é em torno de 200Hz, a frequência de corte (*fc)* deveria ficar em torno de 150Hz, escolheu-se os valores do capacitor e resistor de 1uF e 1kΩ respectivamente obtendo-se o valor da *fc* conforme equação Eq. 17*:*

$$fc = \frac{1}{2\ x\ \pi\ x\ R\ x\ C} \qquad \text{(Eq. 17)}$$

$$fc = \frac{1}{2\ x\ \pi\ x\ 1k\Omega\ x\ 1\mu F} \Rightarrow fc = 159Hz$$

Na Figura 26 mostra o sinal após passar pelas portas inversoras com as suas bordas de subida e de descida abruptas.

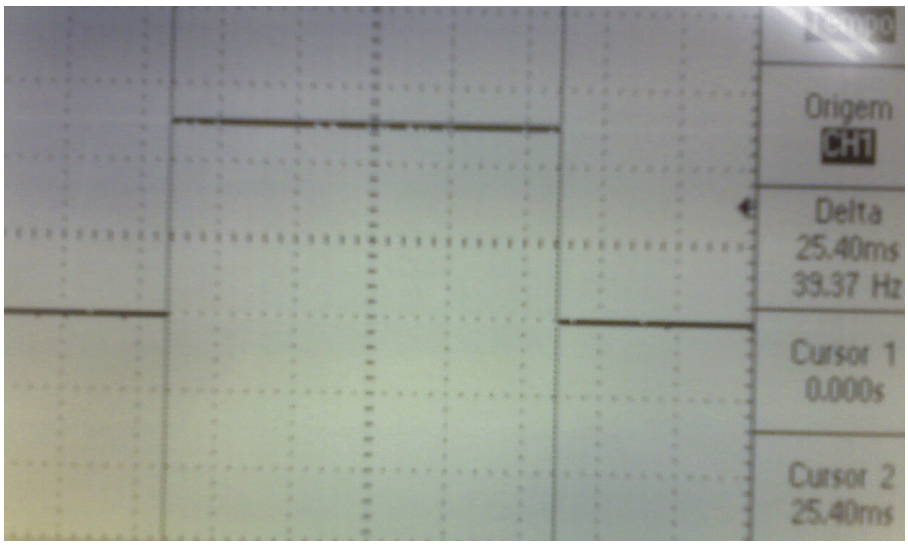


Figura 26 - Sinal do *encoder* após tratamento.

O sinal nos pinos é (0V que representa nível baixo) quando a luminosidade do *encoder* esta bloqueada e quando o disco se encontra na fenda o sinal nos pinos é (5V que representa nível alto).

Na Figura 27 pode ser visto o esquema elétrico dos *encoders*, o circuito inversor e micro-controlador. Os pinos RB4 e RB5 do micro-controlador são utilizados como entradas de interrupção (interrupção ativada nas trocas de estado.

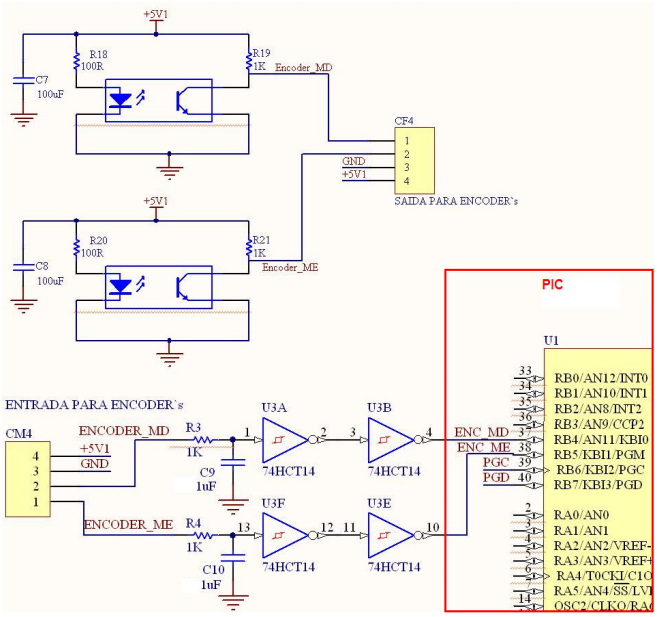


Figura 27 - Esquema elétrico do sistema óptico.

A Tabela 7 mostra a relação entre a porcentagem de PWM e a largura de pulso em nível alto lido pelo micro-controlador. Esta analise foi efetuada em laço

aberto setando-se uma porcentagem de PWM e lendo o tempo que é referente a largura da fenda do roda perfurada do *encoder*.

Tabela 7 - Faixa de PWM e largura de pulso do *encoder*.

PWM (%)	Largura do pulso (MS)
45	23,69
60	15,03
80	9,95
100	7,67

A Figura 28 mostra os componentes roda, *encoder* e roda perfurada e a forma como foram acopladas no protótipo.

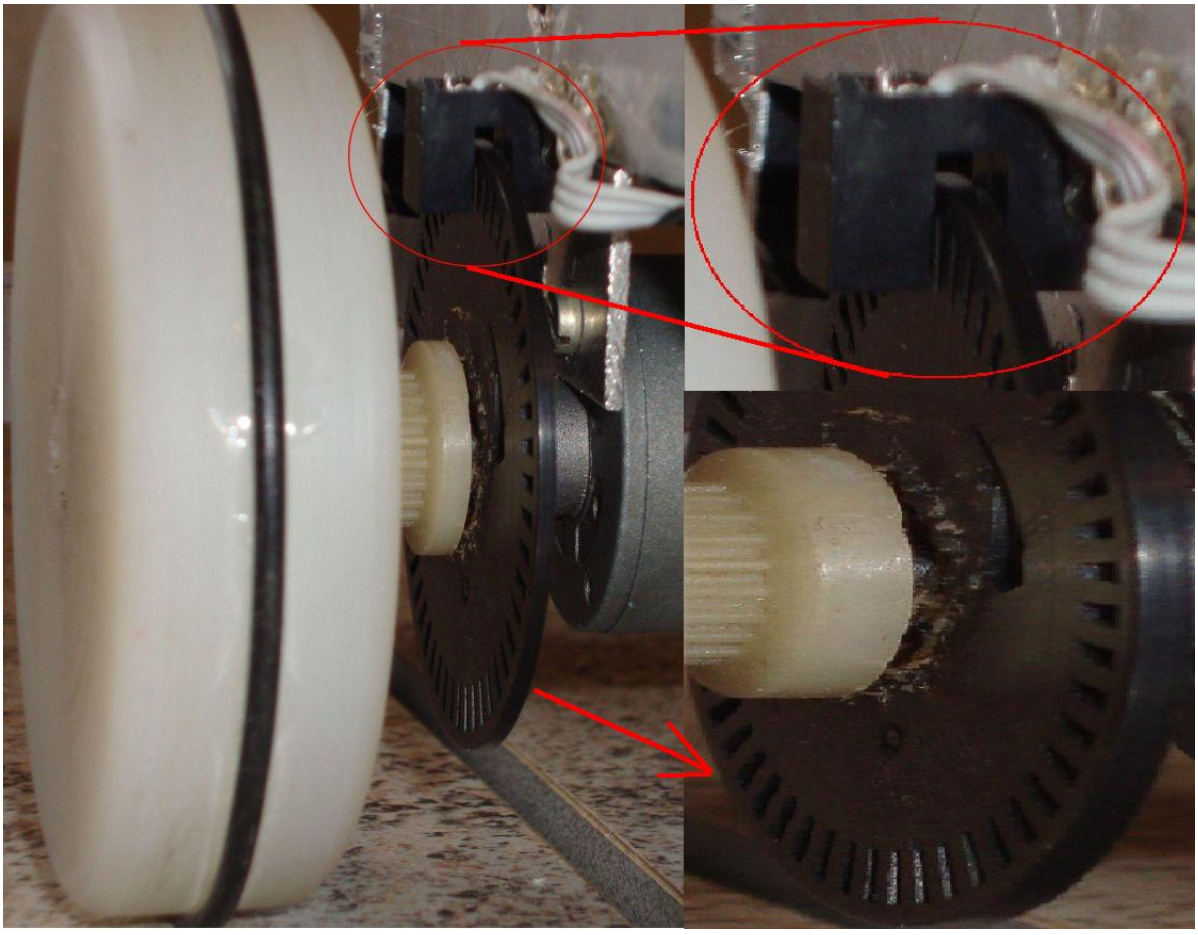


Figura 28 - Detalhe da roda dentada e do sistema óptico acoplado.

Implementação digital da leitura da velocidade

A velocidade é obtida através da leitura do tempo entre as fendas do disco. As velocidades são obtidas individualmente para cada roda com o auxilio de

interrupção externa e *Timer* interno do micro-controlador, o processo de leitura simplificado é: habilita a interrupção externa quando o sinal estiver em nível alto zera e depois habilita o *Timer*, na próxima interrupção para o *Timer,* lê o valor nos bytes alto e baixo e retorna da interrupção com o valor do tempo decorrido, que é utilizado para calcular a velocidade.

Os dados que serão compostos em velocidade são tratados em três funções le_encoder_dir (função que lê a velocidade da roda direita), le_encoder_esq (função que lê a velocidade da roda esquerda) e a *interrupt* (função que trata todos os pedidos de interrupção), abaixo vamos comentar como funcionam as funções le_encoder_esq e *interrupt*.

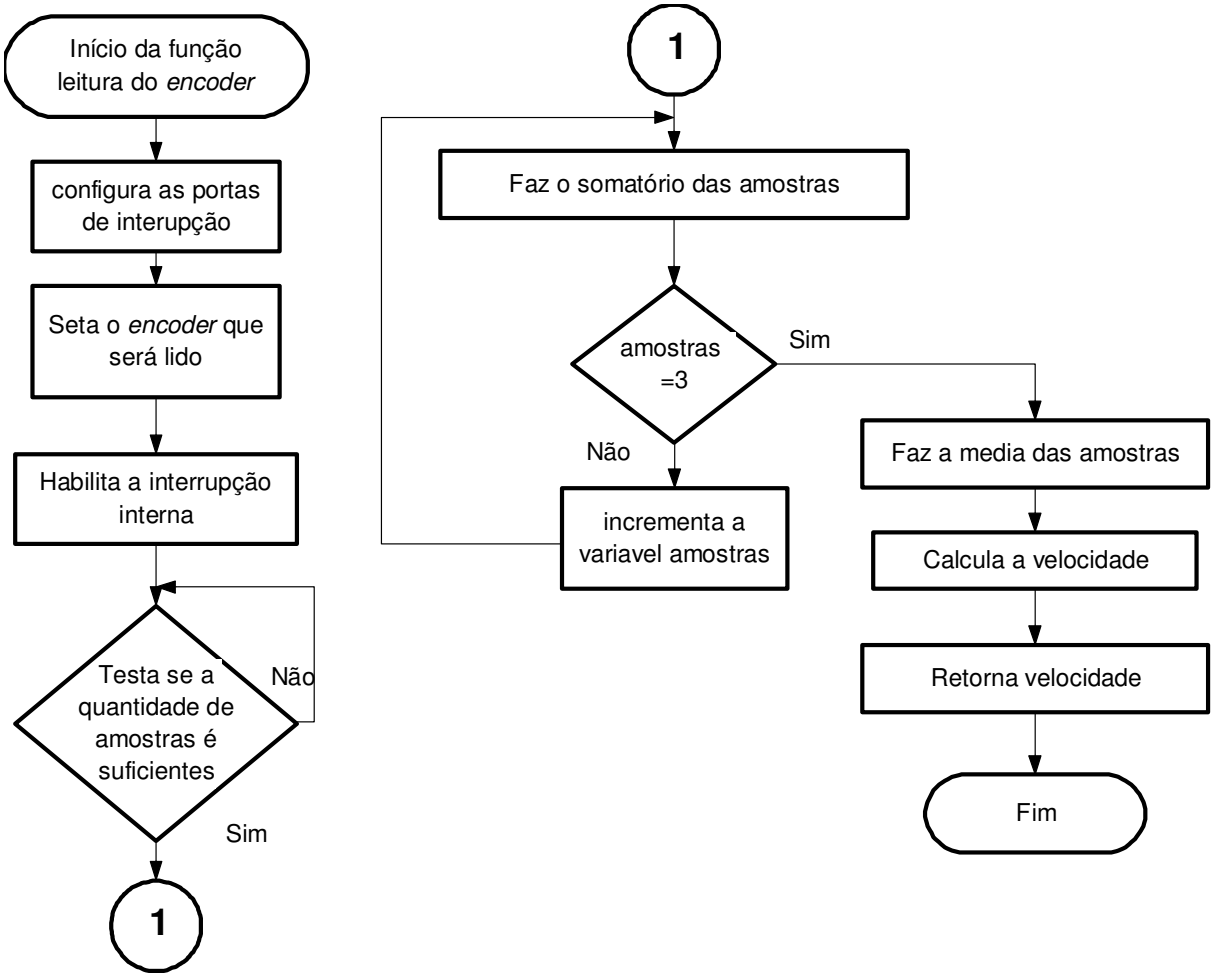


Figura 29 - Fluxograma da função que lê os *encoders*.

A leitura da velocidade é feita individualmente em cada roda, porém é realizado primeiro na roda direita e depois na roda esquerda. Na Figura 29 foi mostrado o fluxograma da função le_encoder_dir e logo a seguir o código do *software* desta função.

int le_encoder_dir()// função que lê a medida da largura do pulso do *encoder* do motor da direita

```
{
int d;
unsigned long soma_rb5 = 0;
   encoder = 2;// seta 2 pra ler o encoder do motor da direita que esta no pinos RB5
   TRISB.RB4  = 0;
   TRISB.RB5  = 1;
   INTCON.RBIE  = 1;// habilita o interrupção
   while (var_vet_rb5 < 4){}// aguarda até que tenha tido 4 amostras
   for (d=2;d<4;d++)// soma as amostras de 2 a 4 = a 3 amostras
   {
     soma_rb5 = soma_rb5 + t_rb5_vet[d];
   }
   t_md_med = (soma_rb5/3);// faz a media, dividindo por 3
   INTCON.RBIE  = 0;// desabilita o interrupção
   var_vet_rb5 = 1;
   TRISB.RB5  = 0;
   compensador = ((pwm_md - 49.42)/181.94);
   velocidade_md = (((1354891/89849)/(t_md_med/3125)) - compensador);
   return velocidade_md; // retorna a média do tempo em que o sinal ficou em nível alto
}// FIM le_encoder_dir
```

A forma de medida de velocidade é idêntica para as duas rodas, portanto não será mostrado o fluxograma para a roda esquerda. O *software* pode ser visto no Apêndice D.

Por característica do micro-controlador todos os pedidos de interrupção dos pinos RB são sinalizados pelo *bit* de *flag* INTCON.RBIF portanto devem ser tratados dentro da mesma função, para verificar qual era o pino de interrupção que deveria ser tratado naquele instante usou-se a técnica de testar qual era o *encoder* que seria lido configurando-se o pino da porta referente a esta interrupção como entrada e o pino da porta da outra interrupção foi configurado como saída, isso garante que o pedido de interrupção é gerado pelo *encoder* certo. Isso possibilitou usar apenas um *Timer* para o dois *encoders,* o valor do ciclo de máquina contado pelo *Timer* é armazenado em um vetor distinto para cada *encoder*.

Na Figura 30 está o fluxograma da função que trata os pedidos de interrupção dos pinos RB4 responsável pela roda direita e RB5 responsável pela roda esquerda e logo a seguir o código do *software* desta função.

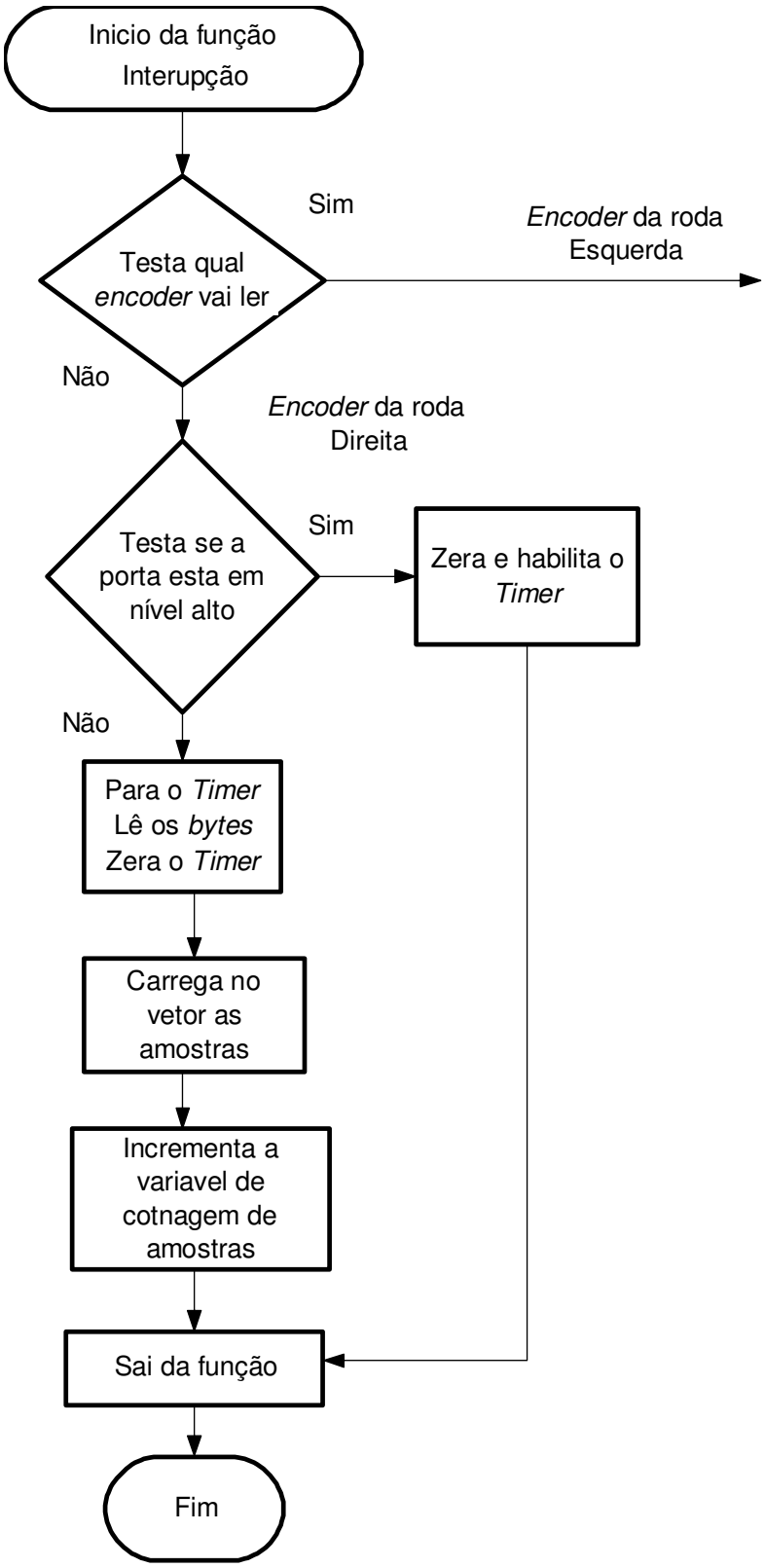


Figura 30 - Fluxograma da função que trata a interrupção externa.

```
void interrupt(void)// função que trata as interrupções.
{
   if(INTCON.RBIF == 1)//trata o interrupção a nos pinos RB4 a 7
   {
   INTCON.RBIF = 0;
    if (encoder == 2)
    {
      if (PORTB.RB5 == 1)// encoder da esquerda
      {
         TMR0L = 0;
         TMR0H = 0;
         T0CON.TMR0ON = 1;
      }
      else
      {
         T0CON.TMR0ON = 0;
         timer0L = TMR0L;// carrega a parte baixa
         timer0H = TMR0H;// carrega a parte alta
         TMR0L = 0;
         TMR0H = 0;
         t_rb5_vet[var_vet_rb5] = ((timer0H << 8) + timer0L);// soma parte alta e a baixa e grava no
vetor
         var_vet_rb5++;
       }
    }
    if (encoder == 1)   // encoder da direita
    {
      if (PORTB.RB4 == 1)//verifica se a transição é de 0 para 1
      {
        TMR0L = 0;
        TMR0H = 0;
        T0CON.TMR0ON = 1;
      }
      else
      {
        T0CON.TMR0ON = 0;
        timer0L = TMR0L;// carrega a parte baixa
        timer0H = TMR0H;// carrega a parte alta
        TMR0L = 0;
        TMR0H = 0;
        t_rb4_vet[var_vet_rb4]= ((timer0H << 8) + timer0L);// soma parte alta e a baixa e grava no vetor
        var_vet_rb4++;
      }
     }
   }
}
```

O código completo das três funções está no Apêndice D.

4. RESULTADOS

4.1 Caracterização das velocidades

4.1.1 Cálculo da relação entre velocidade e PWM

Para calcular a relação entre a velocidade e a porcentagem de PWM foi efetuado o procedimento de acionar os motores com um PWM fixo e obter a velocidade de cada roda, este levantamento foi feito aplicando um valor mínimo de 40% de PWM com incremento de 10% até o valor máximo de 100% de PWM. Caracterizando a relação entre o PWM aplicado e a velocidade lida em cada roda, que pode ser visto nos dados do Gráfico 5.

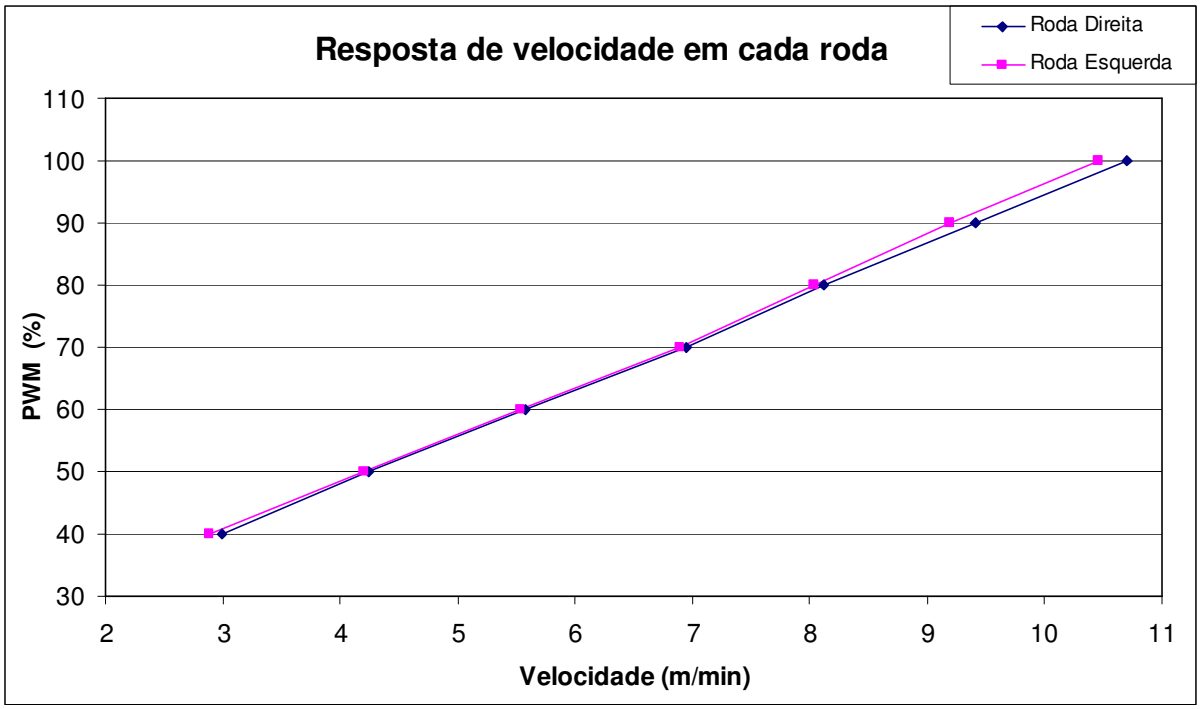


Gráfico 5 - Resposta de velocidade em cada roda.

4.1.2 Correção da velocidade entre as rodas

Conforme pode ser visto no Gráfico 5 há uma diferença entre as velocidades lidas das rodas direita e esquerda, para sanar este problema foi levantado a curva do erro de velocidade entre as rodas e aplicando a correção foi aplicada sobre a roda direita traçando-se uma reta, chamada de reta de tendência, sobre a curva de erro e obtendo-se a equação Eq. 18 mostrado abaixo que relaciona o PWM da roda direita e o erro, diferença das duas velocidades.

$$compensador = \frac{(pwm\_md - 49.42)}{181.94} \qquad \text{(Eq. 18)}$$

Esta constante nomeada de "compensador" é atualizada a cada chamada da função que calcula o PWM das rodas e é diminuída da velocidade da roda direita antes de calcular o novo PWM da roda direita.

Para melhor entender veja o diagrama na Figura 31 com os passos em malha fechada para correção da diferença de velocidade.

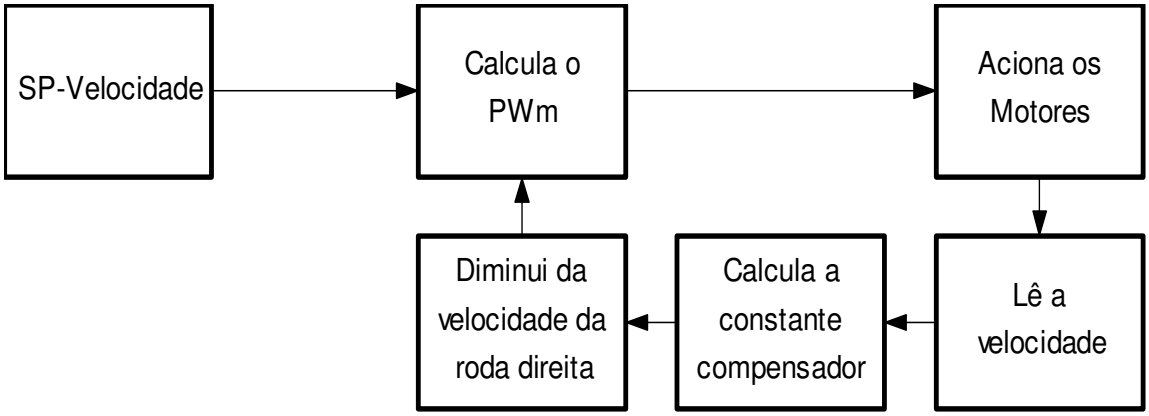


Figura 31 - Diagrama dos passos para correção da diferença de velocidade das rodas.

Após implementar a correção de velocidade descrita na equação Eq. 17 foram refeitos os testes e obteve-se as relações de PWM por velocidade, que é mostrado no Gráfico 6.

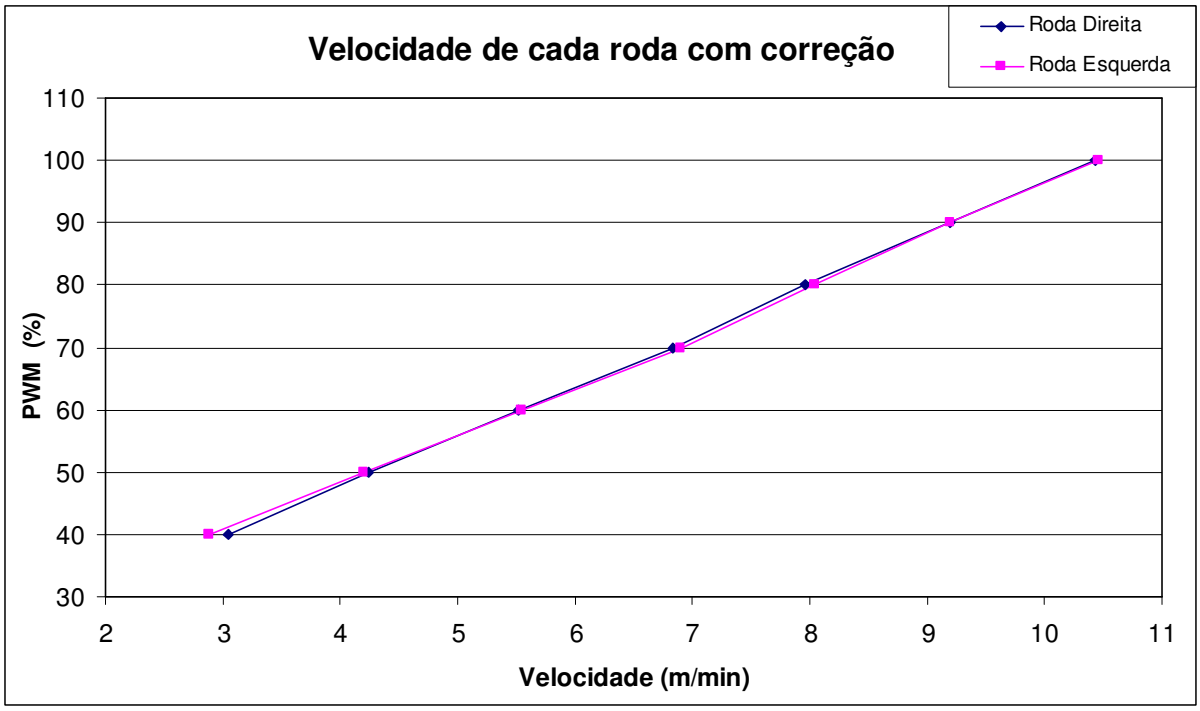


Gráfico 6 - Velocidade de cada roda com correção

A curva padrão utilizada para obter a relação entre velocidade e PWM foi a do motor da direita, foi adicionado sobre esta curva a linha de tendência no qual foram obtidos os valores das constantes de interceptação e inclinação, a equação entre a relação de velocidade e PWM é mostrada na equação Eq.19 abaixo.

$$PWM = ((7{,}82 x Velocidade) + 16{,}5) \qquad \text{(Eq. 19)}$$

4.1.3 Caracterização da velocidade a laço aberto

A implementação de um sistema de controle é possível através do estudo da curva de velocidade resultante do conjunto mecânico completo, com uma taxa de amostragem desta velocidade fixa e conhecida.

Com o sistema de leitura de velocidade instalado no protótipo, que compõem os *encoders*, não foi possível obter estes valores, pois não se tinha uma taxa fixa de amostragem, tendo em vista que a velocidade aumentava gradativamente conforme vencia a inércia e o tempo entre cada pulso do *encoder* ia diminuindo.

A solução aplicada foi a gravação de um vídeo e após converter este vídeo em imagens com tempo fixo entre elas, isto só foi possível pois conhecíamos a

quantidade de fotos por segundo, *Frames* por segundo (FPS) do dispositivo de gravação utilizada para a gravação de vídeos.

O método consiste em gravar o deslocamento do protótipo desde o repouso até a estabilização da velocidade com auxilio de uma maquina digital obtendo um vídeo, que nada mais é que várias fotos tiradas em uma taxa fixa, neste caso de 30FPS resultando em uma foto a cada 33,33ms aproximadamente. Com o auxilio de *softwares* de tratamento de imagens foi possível decompor o vídeo em imagens. Na Figura 32 abaixo é mostrado duas Fotos obtidas da decomposição do vídeo.

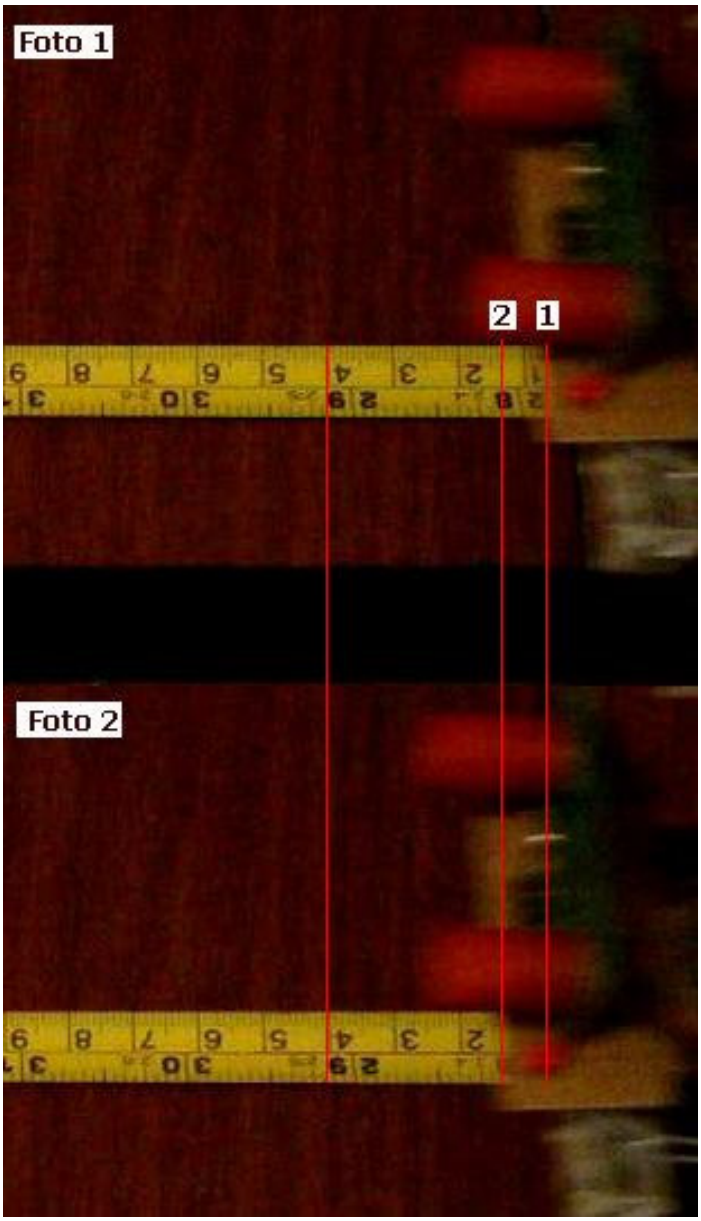


Figura 32 - Duas imagens da decomposição do vídeo em fotos.

Foi utilizada uma trena localizada sob o protótipo para obter a distância exata percorrida em cada tempo fixo, com estas duas informações é possível

determinar a velocidade instantânea em que o protótipo se deslocou desde o repouso até a estabilidade da velocidade.

Para a manipulação dos valores de velocidade no *software MATLAB* 7.0 foi utilizado em m/s, portanto todos os valores foram convertidos de m/min para m/s.

Conforme mostrado na Figura 32 a Foto 1 e dizer que o protótipo está no instante inicial t e distancia d, na Foto 2 podemos dizer que ele esta no tempo final $(t+33{,}33ms)$ e na distancia final de $(d+0{,}7cm)$, com estas informações temos:

$$Vi = \frac{d_{finnal} - d_{inicial}}{t_{final} - t_{inicial}} = \frac{(d+0{,}7cm) - d}{(t+33{,}33ms) - t} = \frac{0{,}5cm}{33{,}33ms} = \frac{0{,}15m}{s} \qquad \text{(Eq. 20)}$$

Onde:

Vi – Velocidade instantânea.

Abaixo podemos ver no Gráfico 7 a curva de velocidade instantânea em uma base de tempo fixa no qual foi utilizado para obter a função de transferência.

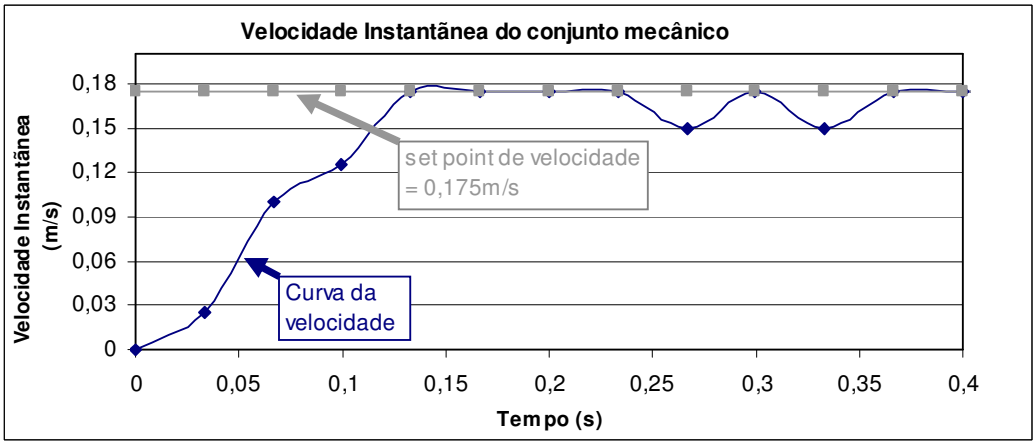


Gráfico 7 - Velocidade instantânea do conjunto mecânico.

4.2 Modelagem obtida a partir do ensaio a laço aberto

O método utilizado para levantamento dos ganhos e a função de transferência é o Método da Resposta ao Salto, modelando o sistema em um sistema de 1ª ordem, Tipo 0, para isso necessitamos conhecer a constante de tempo dominante T que é definida como o tempo que a velocidade instantânea demora para atingir 63% de seu valor total.

O ensaio deve ser feito em laço aberto, os valores obtidos foram importados para o *MATLAB* e podem ser vistos no Gráfico 8.

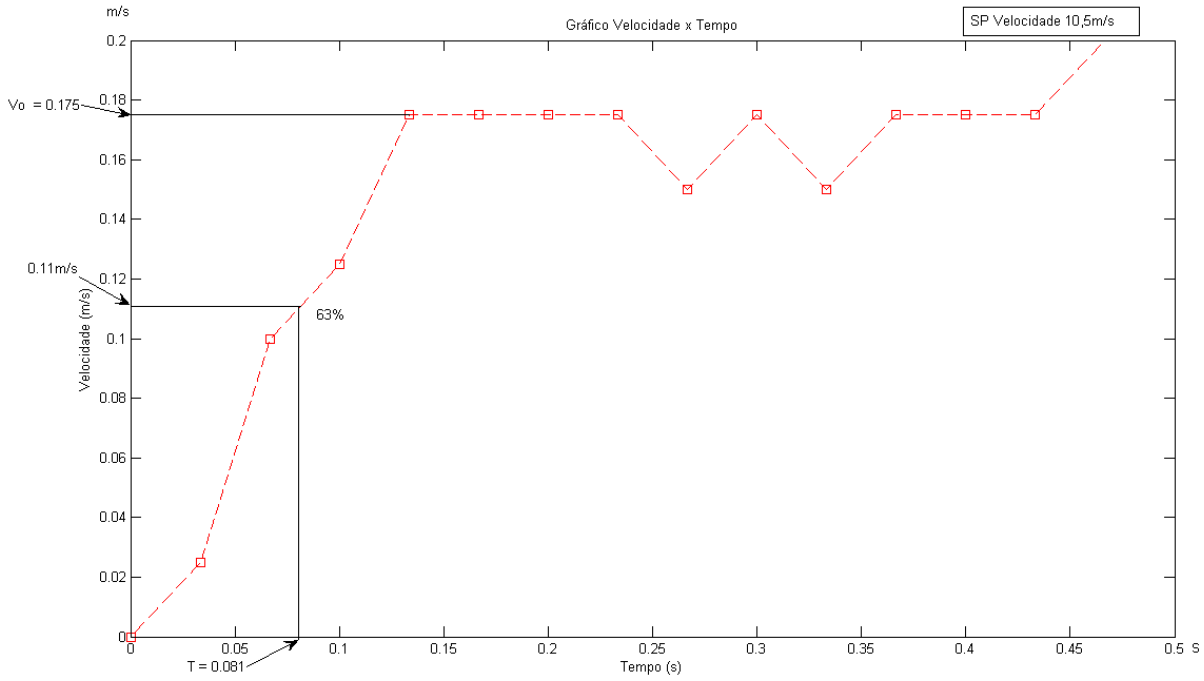


Gráfico 8 - Velocidade instantânea do conjunto mecânico plotado no *MATLAB*.

Do Gráfico 8 têm-se $T = 0{,}081ms$ e $Vo = 0{,}175V$.

Conforme pesquisado na Referência [9] as equações para modelagem de um sistema de 1º ordem são:

$$g(s) = \frac{K}{S + P} \qquad \text{(Eq. 21)}$$

Dos valores de T e Vo têm-se:

$$P = \frac{1}{T} = \frac{1}{0{,}081ms} = 12{,}34 \qquad \text{(Eq. 22)}$$

$$Vo = \frac{K}{P} \Rightarrow K = Vo\ x\ P = 0{,}175\ x\ 12{,}34 = 2{,}16 \qquad \text{(Eq. 23)}$$

Das equações Eq. 21, Eq. 22 e Eq.23 têm-se a função de transferência do sistema g(s):

$$g(s) = \frac{K}{S + P} = \frac{2{,}16}{S + 12{,}34} \qquad \text{(Eq. 24)}$$

Com auxilio do *software MATLAB* 7.0 pode-se obter a curva da função g(s) em malha aberta, onde pode ser visto o amplitude, velocidade em que o sistema estabilizou $Vo = 0{,}175V$ o tempo de estabilização "*Setting Time*" de $Ts = 0{,}249s$ na tolerância de 5%.

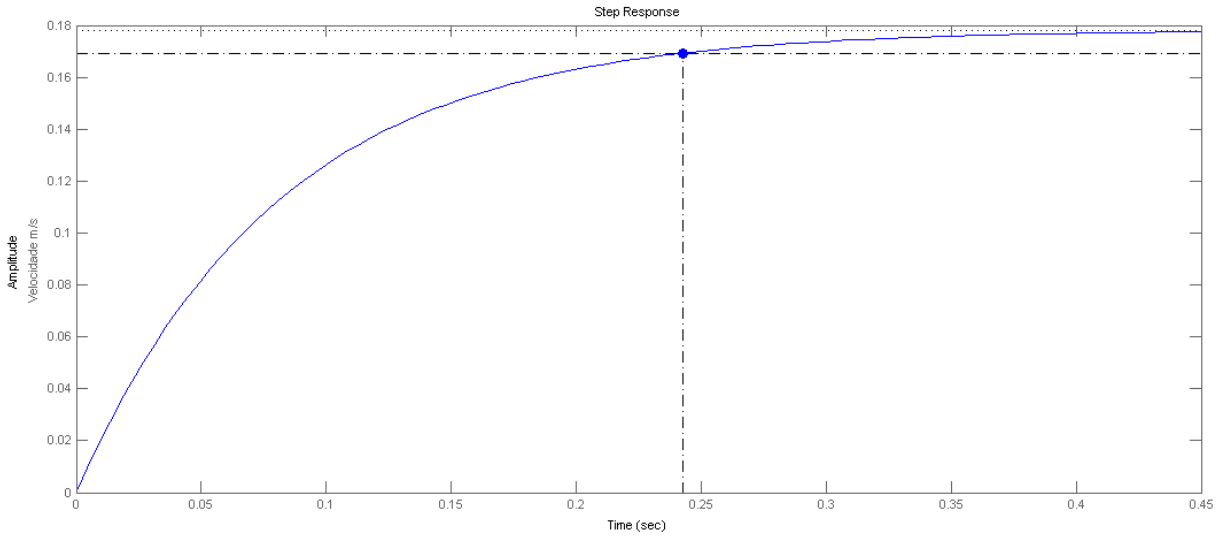


Gráfico 9 - Resposta ao salto unitário do sistema em malha aberta g(s).

No Gráfico 10 abaixo pode ser visto em vermelho a curva com a resposta original do sistema e em azul a curva da resposta da função de transferência obtida através da curva em original do sistema, ambas em malha aberta.

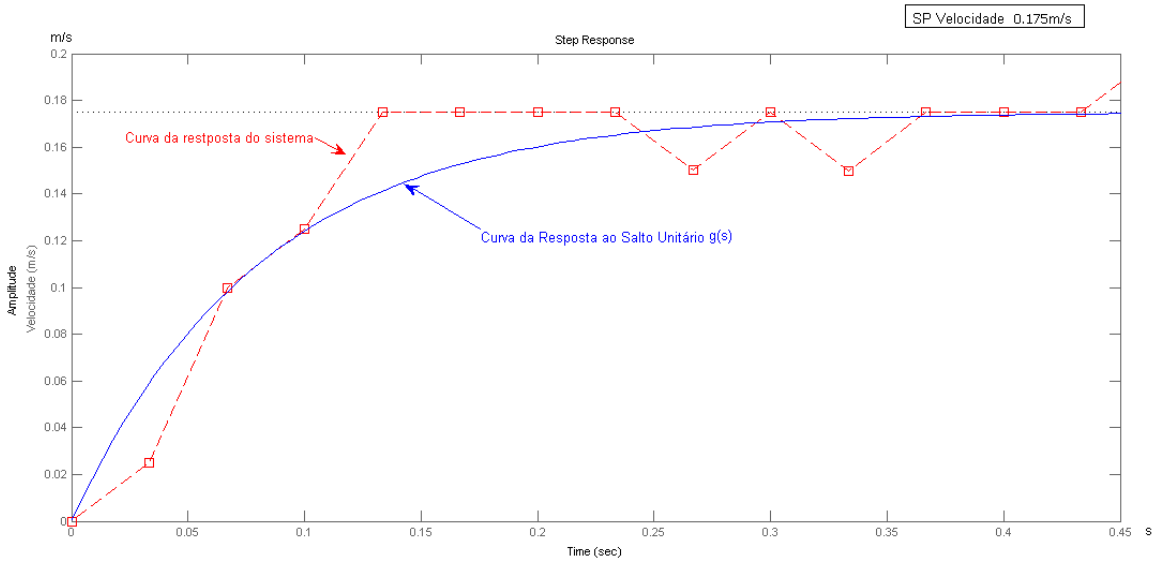


Gráfico 10 - Comparação entre as curvas da resposta do sistema e do salto unitário.

4.3 Estratégia de controle

Foi utilizado um controlador do tipo PID conforme descrito no capítulo 2.4.2. Para se ter um ponto inicial das constantes kp e ki foram projetados e testados sobre a g(s), no *MATLAB*, dois ajustes sendo os controladores P e o PI ainda na forma analógica, para a implementação digital via *software* posteriormente.

No capitulo 2.3.1 estão descritos os dois ajustes P e PI efetuados no *MATLAB* e a seguir no capitulo 2.3.2 a implementação digital destes dois controladores seguindo do fluxograma e o código de *software*.

4.3.1 Ajustes dos controladores P e PI efetuado no *MATLAB*

Ajuste do controlador P

A equação do controlador P é mostrado na Eq. 25.

$$gc(s) = kp \qquad \text{(Eq. 25)}$$

A equação final resultante entre a multiplicação da função de transferência g(s) Eq. 24 e a ação proporcional Eq. 25 é mostrada na equação Eq. 26.

$$gf(s) = g(s)\ x\ gc(s) = \frac{2{,}16}{S + 12{,}34}\ x\ kp = \frac{2{,}16kp}{S + 12{,}34} \qquad \text{(Eq. 26)}$$

A ação P aplicada teve constante $kp = 1.05$ alterando este valor na equação da g(s), Eq. 26 temos.

$$gf(s) = \frac{2{,}16\ x\ 1.05}{S + 12{,}34} = \frac{2{,}268}{S + 12{,}34}$$

A resposta da função de transferência com a ação P pode ser vista no Gráfico 11, na curva em verde e para comparação a curva em azul que é a resposta da função g(s) original.

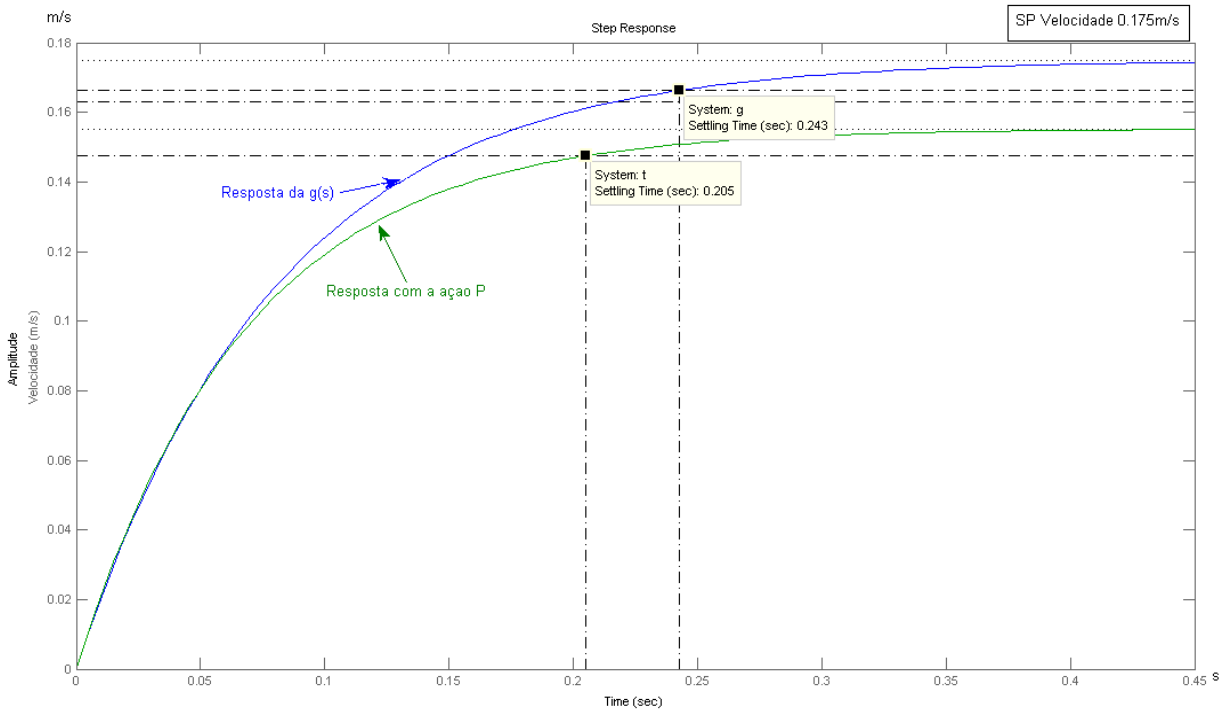


Gráfico 11 - Resposta do sistema com a ação P aplicada.

Ajuste do controlador PI

A equação da ação integral é definida na equação Eq. 27.

$$gc(s) = \frac{ki}{S} \qquad \text{(Eq. 27)}$$

Quando implementado o controlador PI obtém-se a equação definida na Eq.28.

$$gc(s) = \frac{kp\left(S + \frac{ki}{kp}\right)}{S} \qquad \text{(Eq. 28)}$$

Aplicando a ação PI Eq. 28, gc(s), na equação Eq. 24, g(s) resulta na equação final Eq. 29.

$$gf(s) = g(s)\ x\ gc(s)\frac{2{,}16}{S+12{,}34}x\frac{kp\left(S+\frac{ki}{kp}\right)}{S} = \frac{2{,}16kp\left(S+\frac{ki}{kp}\right)}{S(S+12{,}34)} \qquad \text{(Eq. 29)}$$

Os valores de kp e ki foram ajustados na forma analógica no *MATLAB* para obter um Ts em torno de 1 segundo e os valores são $kp = 6.5$ e $ki = 24.375$ aplicando estas constantes na equação Eq. 29 se obtêm:

$$gf(s) = \frac{2{,}16\ x\ 6.5\left(S + \frac{24.375}{6.5}\right)}{S(S+12{,}34)} = \frac{14.04(S+3.75)}{S(S+12{,}34)} \qquad \text{(Eq. 30)}$$

A resposta da função de transferência com a ação PI pode ser vista no Gráfico 12, a curva em preto e para comparação a curva em azul que é a resposta da função g(s) original.

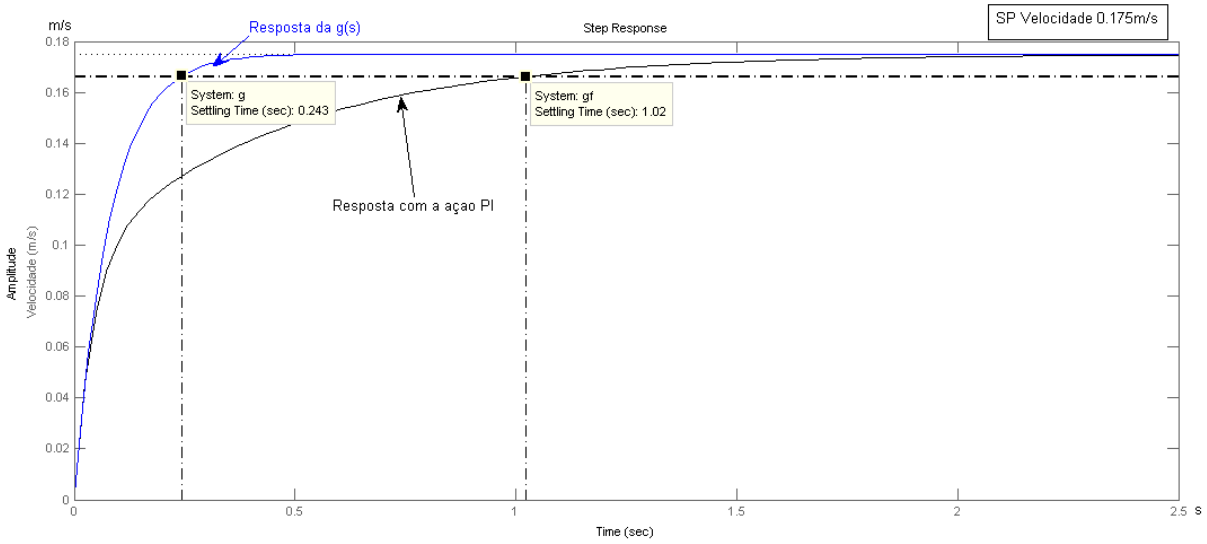


Gráfico 12 - Resposta do sistema com a ação PI aplicada.

4.3.2 Implementação digital dos controladores P e PI

A implementação em *software* dos dois controladores foi realizado dentro da função calcula_pwm(). A programação para os controladores P e PI são praticamente idênticas, o que diferencia é que no controlador P não se tem a ação integral lembrando que a constante kp é diferente para os dois controladores.

Portanto será detalhada com fluxograma e *software* apenas a implementação PI referenciando o que faz parte da ação proporcional e da integral.

Para ilustrar os passos executados na função calcula_pwm() segue fluxograma mostrado na Figura 33. O fluxograma foi simplificado representando apenas para o motor da roda direita sendo idêntica para o motor da roda esquerda. No inicio da função são atribuídas as constantes kp e ki seguindo dos cálculos da ação PI e o cálculo relacionando a ação em PWM para a roda direita.

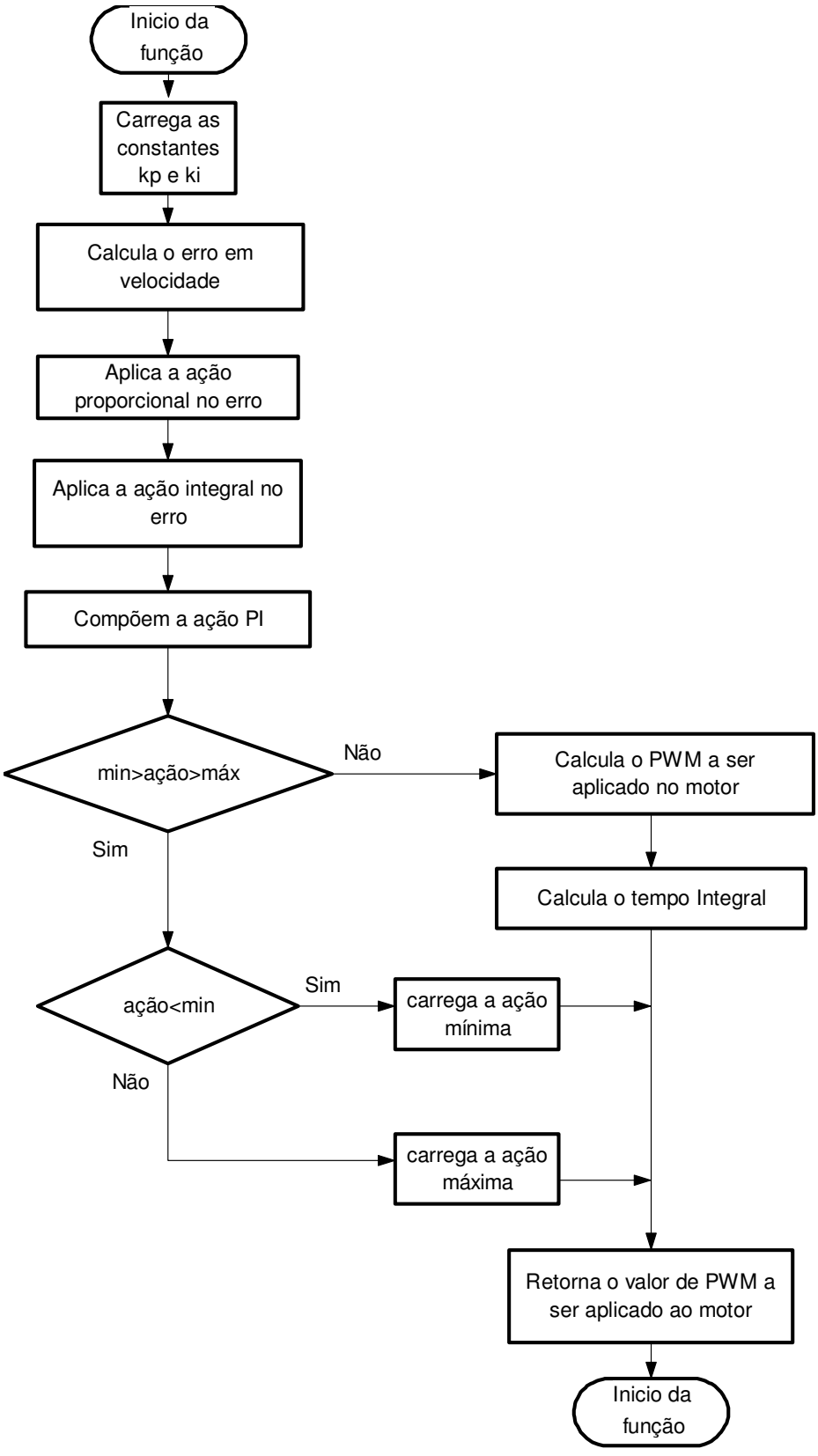
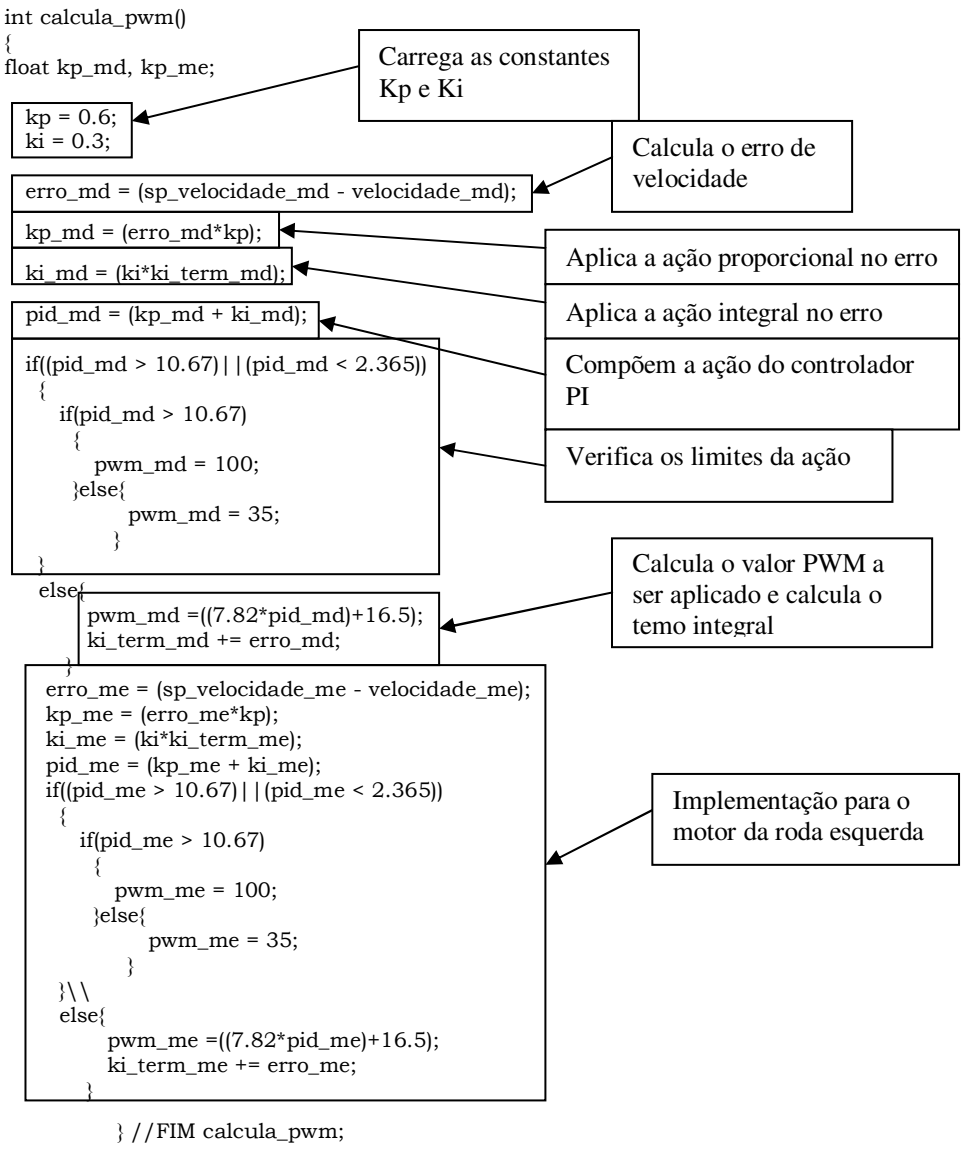


Figura 33 - Fluxograma da função de controle com a ação PI.

Após os cálculos da velocidade com a ação PI já aplicada foi necessário testar este valor, limitando em mínimo de 2.365 e máximo de 10.67, valores estes

que aplicados na equação Eq.18 convertem nos valores mínimos e máximos de 35% e 100% respectivamente de porcentagem do PWM a ser aplicado aos motores. Estes valores mínimos e máximos foram estabelecidos para gerar a saturação, pois após os testes ficou constatado que aplicando valores inferiores a 35% de PWM nos motores o protótipo não vencia a inércia mecânica e os valores maiores que 100% não são aceitáveis.

Abaixo pode ser visto o *software* que implementa o controlador PI, onde são destacados os itens principais mostrados no fluxograma da Figura 33.



Os valores de PWM retornados da função descrita acima, que serão aplicados nos motores através da função aciona_motor, o código referente a esta função pode ser visto no Apêndice D.

As equações Eq. 31 à Eq. 35 mostradas abaixo foram retiradas do trecho do código da função acima e mostrado a seguir com a descrição de cada termo.

$$erro\_md = (sp\_velocidade\_md - velocidade\_md) \quad \text{(Eq. 31)}$$

$$kp\_md = (erro\_md * kp) \quad \text{(Eq. 32)}$$

$$ki\_md = (ki\_term\_md * ki) \quad \text{(Eq. 33)}$$

$$pid\_md = (kp\_md + ki\_md) \quad \text{(Eq. 34)}$$

$$ki\_term\_md = (ki\_term\_ant\_md + erro\_md) \quad \text{(Eq. 35)}$$

Onde:

erro_md – erro de velocidade motor da direita do motor da direita;

ki - constante da ação integral;

ki_md – parcela da ação integral – I do motor da direita;

ki_term_ant_md – valor anterior da ação integral do motor da direita;

ki_term_md – valor atual da ação integral do motor da direita;

kp - constante da ação proporcional .

kp_md – parcela da ação proporcional - P do motor da direita;

sp_velocidade_md – *set point* de velocidade do motor da direita;

velocidade_md – velocidade instantânea do motor da direita;

O valor do pid_md definido anteriormente na equação Eq. 34 é um valor de velocidade que é aplicado na equação Eq.19 implementada para a roda direita, resultando no valor de PWM a ser aplicado ao motor.

4.4 Resultados do sistema controlado

A seguir serão mostrados os resultados obtidos da velocidade com as ações de controle P e PI aplicadas na função de transferência g(s) projetada e descrita no neste capitulo.

4.4.1 Resultado da ação proporcional - P

O sistema foi modelado como sendo de 1° ordem como já comentado, conforme a Referência [9] ao aplicar um degrau unitário em uma função do Tipo 0 resulta um erro em regime permanente (erp), que pode ser calculado através da equação Eq. 36 descrita abaixo.

$$erp = \frac{1}{1+kp} \qquad \text{(Eq. 36)}$$

Com a constante $kp = 1.05$ aplicada na equação Eq. 36 temos o erro teoricamente de:

$$erp = \frac{1}{1+1,05} = 0,487 \Rightarrow ou \Rightarrow 48,7\%$$

No Gráfico 13 abaixo pode ser visto a reta em cinza representando o *set point* de 0,114m/s (6,84m/min que representa 70% de PWM) e a curva de velocidade em azul com a instabilidade da velocidade até aproximadamente 0,8 segundos e após a estabilidade em torno de 0,061m/s que é 46,5% de erro em erp.

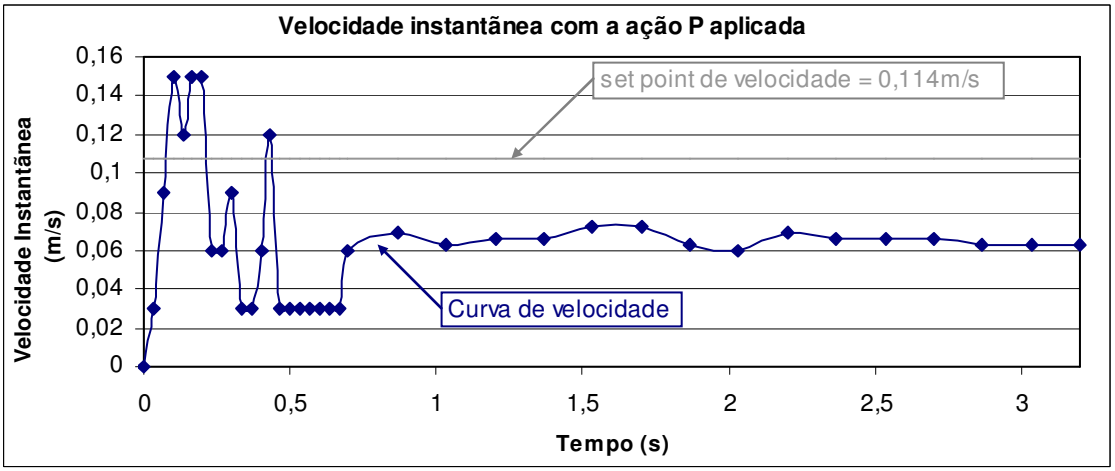


Gráfico 13 - Velocidade instantânea com ação P aplicada.

Como este erro já era esperado aplicando a ação P no sistema modelado, partiu-se para o controle aplicando a ação PI cujos resultados serão descritos na continuação.

4.4.2 Resultado da ação proporcional integral – PI

Ao aplica uma ação PI em uma função g(s) por característica aumenta-se o Tipo, neste caso que o sistema foi modelado de 1° ordem Tipo 0 passa a ser Tipo 1, conforme a Referência [9] ao aplicar um degrau unitário em uma função do Tipo 1 o erro erp é nulo. Ao programar o PI, via *software*, com os valores de $kp = 6.5$ e $ki = 24.375$ obtidos a partir do ajuste feito no *MATLAB* mostrado anteriormente, observou-se que o sistema ficava instável em toda a escala de velocidade e foi necessário alterar estes valores.

Foi fixado o *set point* em 0,114m/s (6,84m/min) que resulta em 70% da ação em PWM aplicado aos motores, foi escolhido este valor para teste por ser o meio da escala, após alguns testes obteve-se os valores das constantes $kp = 0.6$ e $ki = 0.3$ que atendiam as especificações de erp nulo e Ts em torno de 1 segundo. Com estes valores a faixa de velocidade para atuação foi reduzida garantindo a estabilidade, a faixa manteve-se em 0,0716m/s e 0,1666m/s (4,3m/min e 10m/min).

Podemos ver no Gráfico 14 no detalhe a curva em azul, o erro em erp é nulo e o tempo de estabilização em torno de 1,2 segundo.

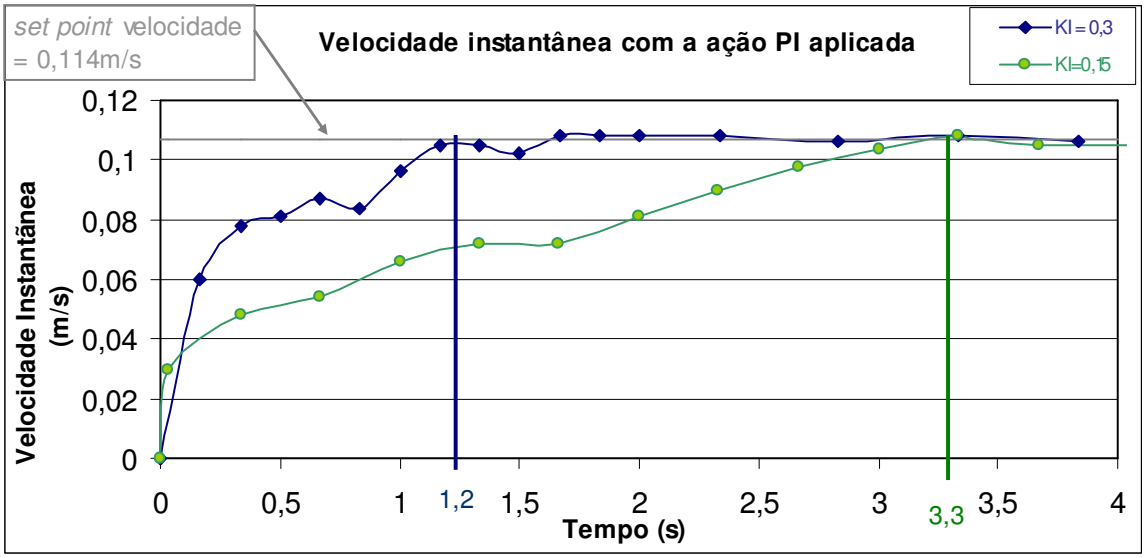


Gráfico 14 - Velocidade instantânea com a ação PI aplicada.

Para analisar o comportamento do sistema alterou-se o valor da constante $Ki = 0.15$ e manteve-se a constante $Kp = 0.6$ e observou-se que o tempo de estabilização ficou em torno de 3,3 segundos no detalhe a curva está em verde.

4.5 Erros observados

4.5.1 Erro de velocidade

Para verificarmos se a resposta de velocidade programada via *set point* estava coerente foram realizadas medidas da seguinte forma, setou-se um *set point* de velocidade e com o veículo em movimento numa distância conhecida mediu-se o tempo decorrido obtendo-se a velocidade, repetiu-se este teste 5 vezes para se ter a média, o teste foi efetuado para cada faixa de velocidade conforme podemos ver na Tabela 8 abaixo.

Tabela 8 - *Set point* de velocidade e velocidade medida.

Set point de velocidade (m/min)	Velocidade medida (m/min)	Desvio padrão (m/min)	Erro (%)
4,3	4,89	0,07	13,66
6	6,54	0,07	9,07
8	8,51	0,32	6,40
10	10,49	0,13	4,92

O erro entre o *set point* e a velocidade medida é mostrado no Gráfico 15.

Idealmente a velocidade setada é a velocidade medida, logo segue uma reta velocidade medida = *set point,* esta reta foi mostrada em cinza. Porém nesse caso há um erro que é devido ao tempo de estabilidade do sistema pela ação PI. O erro é devido á forma de leitura da velocidade comentada no capitulo 3.6 e este aumenta conforme a velocidade diminui.

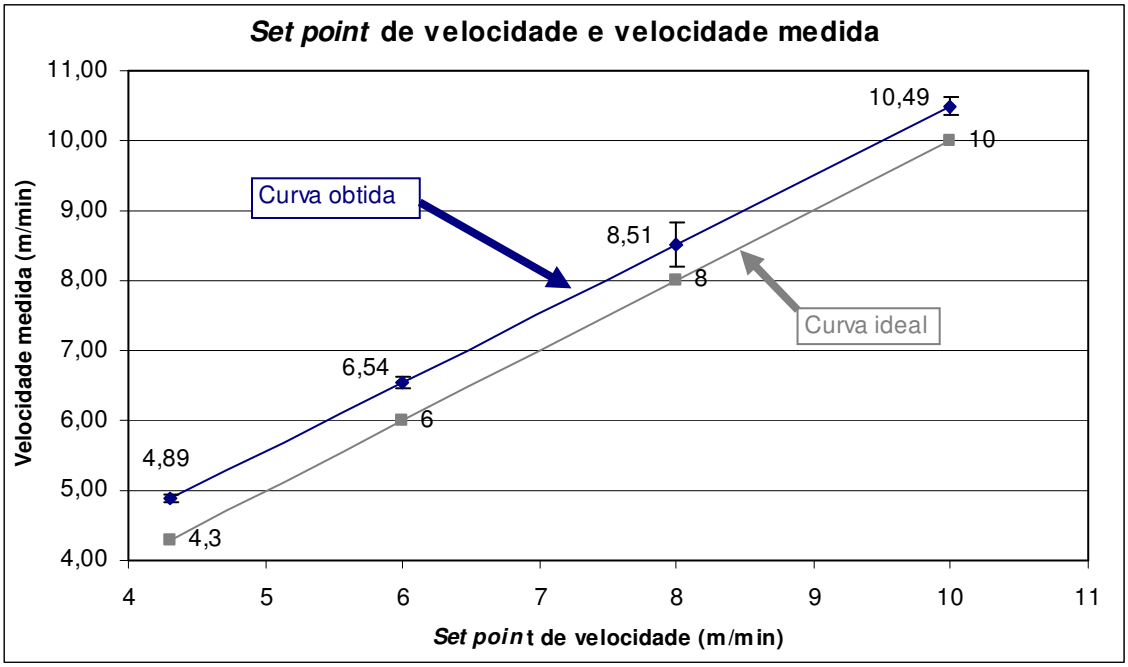


Gráfico 15 - *Set point* de velocidade e velocidade medida.

Como é feita a media de 3 valores do tempo lido referente ao espaçamento da fenda, nas velocidades menores este tempo é maior, isto caracteriza o sistema ser mais lento para ler uma nova velocidade e aplicar a ação correspondente.

4.5.2 Erro da trajetória

Ao setarmos um *set point* de velocidade constante, tendo o *feedback* de velocidade e calculando a ação PI independente para cada roda, esperava-se que o protótipo andasse em uma trajetória reta. Na pratica, não foi o que aconteceu e a trajetória é desviada de uma linha reta.

Para entendermos melhor o que está acontecendo segue abaixo a representação do erro da trajetória na Figura 34.

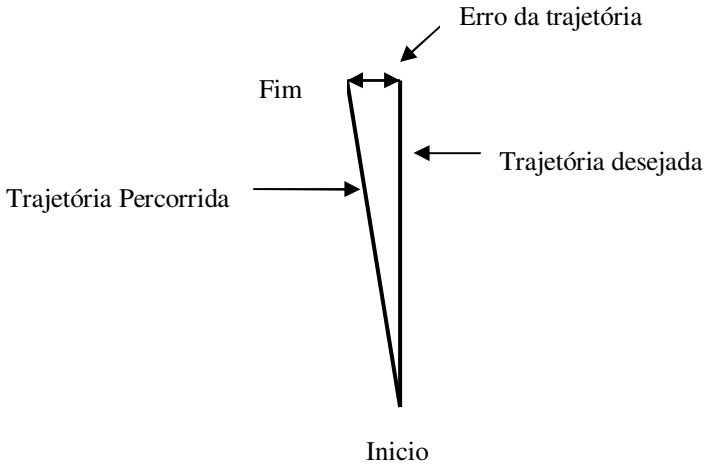


Figura 34 - Representação do erro da trajetória.

Foram realizados 5 ensaios para cada *set point* de velocidade configurado, obtendo a média no erro da trajetória, que é mostrado na Tabela 9 abaixo onde pode ser visto a velocidade setada com o erro em centímetros da trajetória

Tabela 9 - *Set point* de velocidade e erro da trajetória.

Set point de velocidade (m/min)	Distância em linha reta (m)	Distância percorrida (m)	Erro da trajetória (cm)
4,3	6	6,0019	19
6	6	6,0019	15
8	6	6,0081	31
10	6	6,0025	17

Esse erro na trajetória é devido a dois fatores, o primeiro ocorre devido ao cálculo de compensação da velocidade descrito anteriormente com a equação Eq. 17, que depende do PWM atual. O segundo fator, tendo em vista a estratégia usada para a leitura da velocidade é lenta se houver um desvio da trajetória entre estas leituras para o sistema é transparente portanto não será tomada nenhuma ação para corrigir este desvio.

5. CONSIDERAÇÕES FINAIS

5.1 Conclusões

O controle de velocidade é essencial para esta aplicação e o desenvolvimento do protótipo teve um resultado satisfatório, tendo em vista que os motores utilizados têm torque suficiente para estabilizar a velocidade setada em torno de 0,2s sem controle. A utilização dos *encoders* com pouca resolução não contribuiu, dificultando inclusive a caracterização do protótipo em função da velocidade.

A implementação do controle proporcional como esperado para um sistema de 1º ordem do Tipo 0 levou o sistema a apresentar um erro em erp na prática de aproximadamente 46,5% ficando próximo ao calculado que foi 48,7%.

A implementação do controle proporcional integral com as constantes kp e Ki ajustadas na forma analógica com o *MATLAB* na implementação digital não teve o resultado esperado, levando o sistema á instabilidade. Porém ao reajustar os valores observaram-se resultados satisfatórios, mantendo o tempo de estabilização para a velocidade de *set point* de 6,48m/min (70% de PWM) em torno de 1,2 segundos.

Para se obter uma performance melhor na leitura da velocidade a substituição dos discos perfurados por outros com maior resolução é imprescindível, com isso aumentaria a precisão na forma de medida do sistema. A substituição dos motores por outros com menos redução é possível atingir velocidades superiores. Melhorar a interface de entrada adicionado um teclado para os usuários digitarem o *set point* de velocidade. Integração com o sistema de controle de posição e detecção de obstáculos.

6. REFERÊNCIAS

[1] ALVES, JOSÉ L.L., Instrumentação, Controle e Automação de Processos, LTC, 2005;

[2] BALBINOT, ALEXANDRE, BRUSAMARELLO, VALNER JOÃO. Instrumentação e Fundamentos de Medidas, Volume 1, LTC, 2006;

[3] DEL TORO ,VICENT, Fundamentos de Máquinas Elétricas, LTC, 1994;

[4] IOVINE, JOHN. Robots, Androids and Animatrons. 1 ed. McGran-Hill, 1997 ISBN 0-07-032804-8;

[5] FERRARI, M., FERRARI, G., HEMPEL, R., Building Robots with Lego Mindstorms. 1 ed. Syngress, USA, 2002 ISBN 1-928994-67-9;

[6] JONES, J. L., SEIGER, B. A., FLYNN, A.M. Mobile Robots. 2 ed. AK Peters, Natick Massachusetts, 1998 ISBN 1-56881-097-0;

[7] MARTIGNONI, AFONSO, Máquinas Elétricas de Corrente Contínua, 3 ed, POA, 1997;

[8] NICOLOSI, D. E. C. Microcontrolador 8051 Detalhado. 1 ed. São Paulo: Érica, 2000;

[9] NISE, NORMAN S. Engenharia de Sistemas de Controle. 3ed, LTC, 2002 ISBN 8521613016.

[10] OGATA, KATSUHIKO – Engenharia de Controle Moderno – 4a. Ed. – Sao Paulo, Editora Prentice Hall do Brasil, 2005;

[11] SÉRIE BRASILEIRA DE TECNOLOGIA, Seleção de Máquinas Elétricas, v1, Siemens S.A, 1988;

[12] SONNINO, SÉRGIO, Mecânica Geral, Cinemática e Dinâmica, Livraria Nobel S.A. 3 ed, 1985;

SITES CONSULTADOS

[13] Wikipédia – http://en.wikipedia.org/wiki/Automated_Guided_Vehicle

http://en.wikipedia.org/wiki/File:Pid-feedback-nct-int-correct.png

Acessado em 07/06/2009;

[14] Feira de Ciências - http://www.feiradeciencias.com.br

Acessado em 07/06/2009;

[15] Netrino-http://www.netrino.com/Embedded-Systems/How-To/PWM-Pulse-Width-Modulation

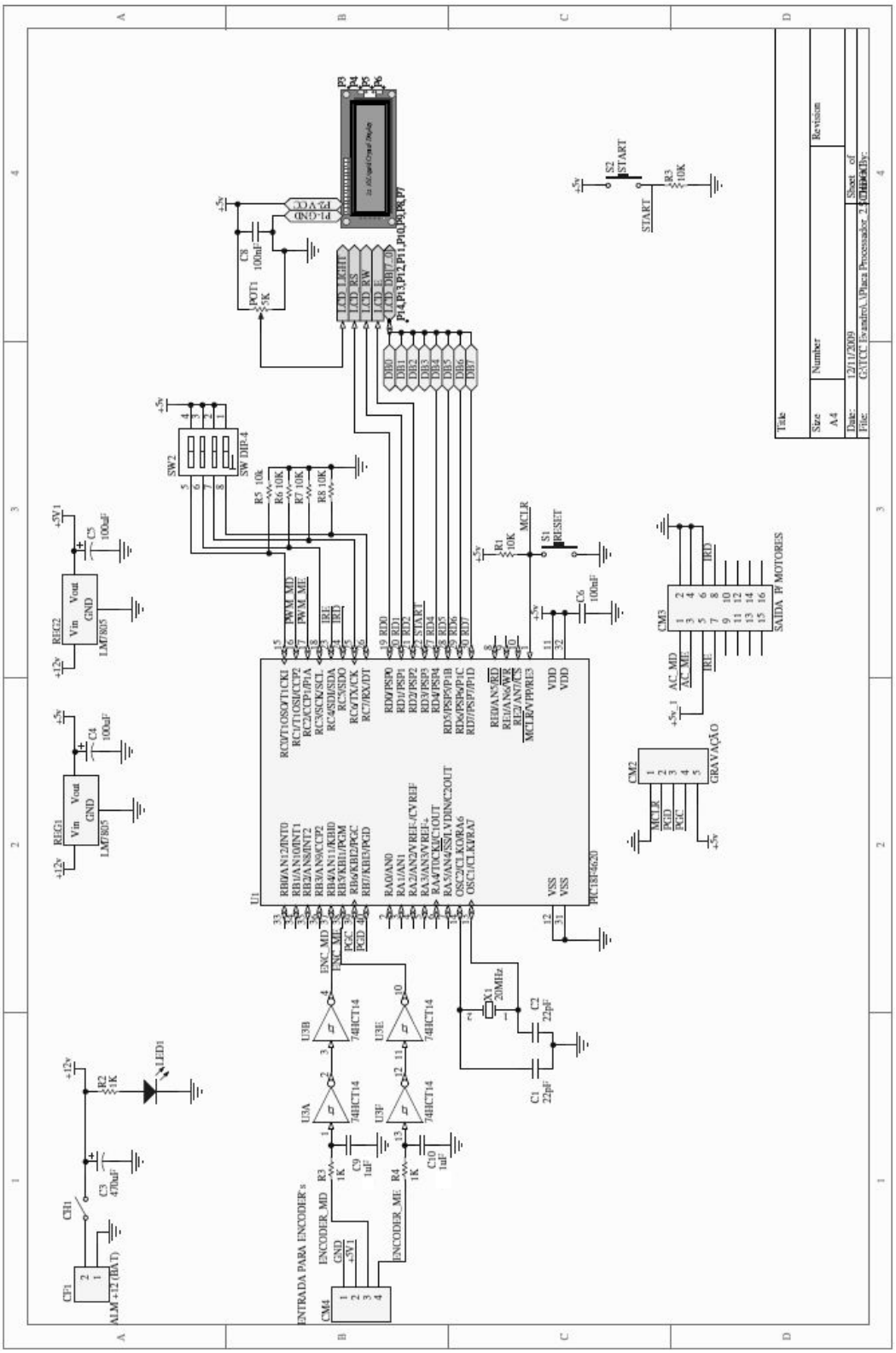
Acessado em 10/06/2009;

OBRAS CONSULTADAS

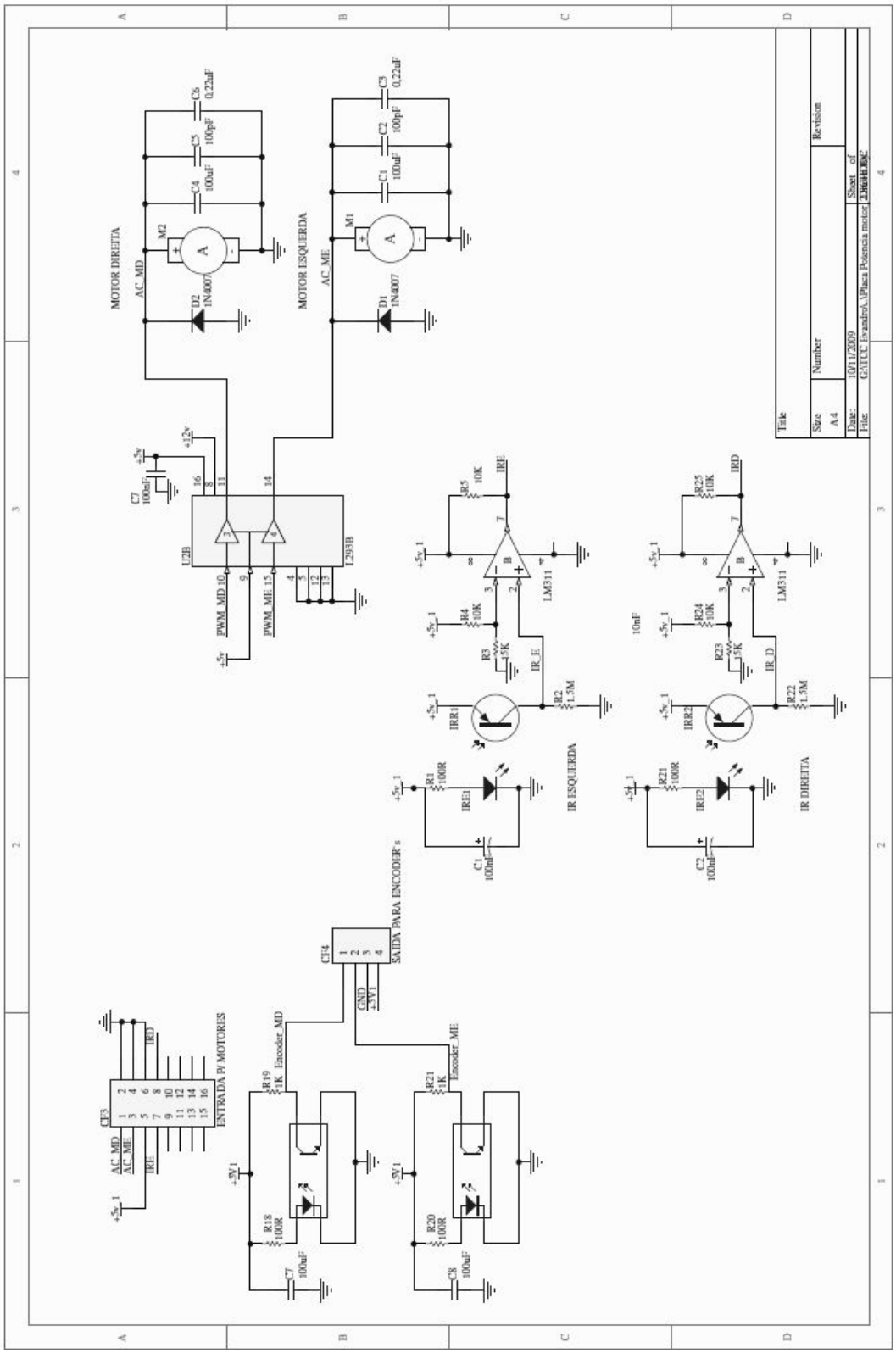
[16] Brusamarello, Valner – Utilização de sensores e transdutores ópticos, princípios de funcionamento – Tipos e Aplicações;

[17] Fischer , Fabio de Oliveira – Controle de Velocidade de Motor Brushless DC, 2008/2.

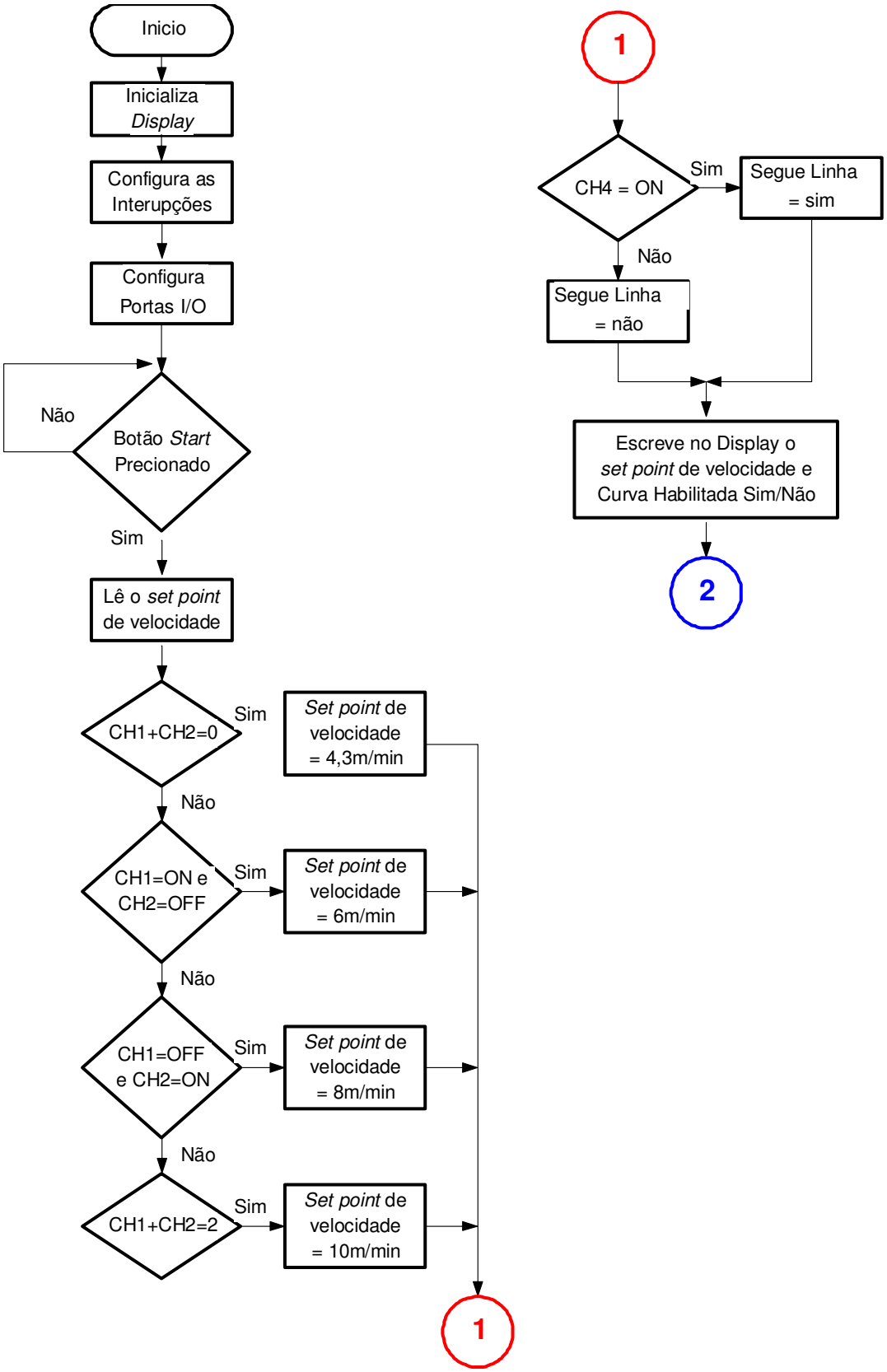
APÊNDICE A – ESQUEMA ELÉTRICO DO CIRCUITO DA CPU

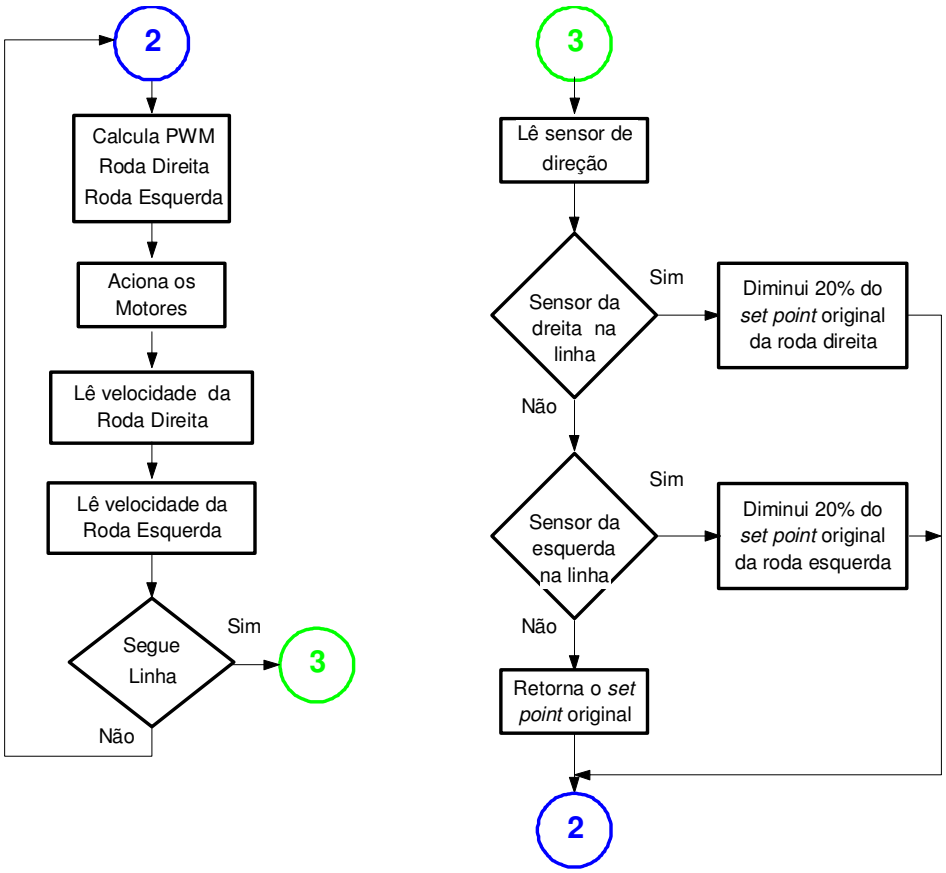


APÊNDICE B – ESQUEMA ELÉTRICO DO CIRCUITO DE POTÊNCIA



APÊNDICE C – FLUXOGRAMA DO *SOFTWARE*




2
Calcula PWM
Roda Direita
Roda Esquerda
Aciona os Motores
Lê velocidade da Roda Direita
Lê velocidade da Roda Esquerda
Segue Linha
Sim
Não
3
3
Lê sensor de direção
Sensor da dreita na linha
Sim
Diminui 20% do set point original da roda direita
Não
Sensor da esquerda na linha
Sim
Diminui 20% do set point original da roda esquerda
Não
Retorna o set point original
2

APÊNDICE D – *SOFTWARE*

```
/*************************************************************************
DECLARAÇÃO DAS FUNÇÕES
*************************************************************************/
int start_button(); // botão de START em RD3
void interrupt();// função das interrupções, obs.
int ini_interrupt();  // configura as interrupções.
int setup_io();   // configura a inicialização das portas de I/O
int ini_lcd();   // configura a inicialização do display
int le_encoder_dir(); // lê o encoder da direita e retorna o tempo em alto.
int le_encoder_esq(); // lê o encoder da esquerda e retorna o tempo em alto.
int ini_pwm(); // configura os pwms
int aciona_motor();// Aciona os motores com o pwm desejado
int calcula_pwm();//calcula o pwm sobre a diferença entre os motores
int le_velocide();//le a velocidade escolhida pelo usuário via dip
int le_sensor_direcao();// lê os sensores IR da direita e da esquerda
int seta_direcao();
int stop();
int stop();
int constante_curva();
/*************************************************************************
DECLARAO DAS VARIAVEIS GLOBAIS
*************************************************************************/
unsigned timer0H, timer0L = 0;
unsigned t_rb4_vet [5]; // vetor para alocar as medidas para media
unsigned t_rb5_vet [5]; // vetor para alocar as medidas para media
float t_md_med = 0; // variável onde está a media do tempo de 3 medidas
float t_me_med = 0; // variável onde está a media do tempo de 3 medidas
int encoder = 0; // digo qual ecoder ele vai ler ( se é 0 vai ler o da direita se for 1 vai ler o da esquerda)
int var_vet_rb5 = 1; // variável do vetor, inicia com 1 e termina com 5
int var_vet_rb4 = 1;
unsigned short pwm_md = 0, pwm_me=0; // variáveis com o valor de pwm de 0 a 100%
char imp_variavel[4]; // variável de 4 posições
float velocidade_md = 0.0;
float velocidade_me = 0.0;
float erro_md = 0.0;
float erro_me = 0.0;
float compensador = 0.0;
float compensador_me = 0.0;
float sp_velocidade = 0.0; // em metros por minuto
float sp_velocidade_md = 0.0; // em metros por minuto
float sp_velocidade_me = 0.0; // em metros por minuto
float sp_velocidade_md_new = 0.0; // em metros por minuto
float sp_velocidade_me_new = 0.0; // em metros por minuto
float pid_md, pid_me;
float ki_term_md = 0.0, ki_term_me = 0.0;
float ki_md, ki_me;
float kp = 0.0;
float ki = 0.0;
float metros_curv_dir = 0.6; //metros para chegar a curva OBS. valor inserido pelo usuário
float tempo;// tempo em segundos para chegar na curva
int sensor = 0; // variável usada para indicar qual sensor leu a fita da direçã0
int segue_linha = 0;
float constante_lo;
float constante_hi;
float constante_lo_new;
float constante_hi_new;
int cont_loop;
/*************************************************************************
INICIO DO PROGRAMA
*************************************************************************/
```

```
void main()
{
    ini_pwm();
    ini_lcd();
    setup_io();
    ini_interrupt();
    Lcd_Custom_Out(1, 1, "START -> INICI0");     // Escreve no display (linha, coluna).
    Lcd_Custom_Out(2, 1, "REV_2 18/11/09");
    while (!start_button());         // aguarda a tecla START ser pressionada
    Lcd_Custom_Cmd(LCD_CLEAR); // Limpa display
    le_velocide(); // le a velocidade setada pelo usuário
    Lcd_Custom_Cmd(LCD_CLEAR);
    Lcd_Custom_Out(1, 1, "Veloc: ");     // Escreve no display (linha, coluna).
    FloatToStr(sp_velocidade, imp_variavel);// converte Valor decimal para sring e imprimir no display
    Lcd_Custom_Out(1, 8, imp_variavel);// imprime no display
    Lcd_Custom_Out(1, 12, "m/min");     // Escreve no display (linha, coluna).
    Lcd_Custom_Out(2, 1, "Segue Linha: ");     // Escreve no display (linha, coluna).
    if(segue_linha == 1)
     {
       Lcd_Custom_Out(2, 12, "Sim");     // Escreve no display (linha, coluna).
     }else
         {
           Lcd_Custom_Out(2, 12, "Nao");     // Escreve no display (linha, coluna).
         }
     while (1)
     {
       calcula_pwm(); // calcula o pwm  de cada roda
       aciona_motor(); // aciona os motores
       le_encoder_dir();
       le_encoder_esq();
       if(segue_linha == 0)
       {
         le_sensor_direcao();
         seta_direcao();
       }
    }// fim while
} // FIM DA MAIN
/*****************************************************************************
FUNÇÕES
*****************************************************************************/
int seta_direcao()
{
          if(sensor == 1) //Sensor da direita na linha
            {
              if (cont_loop > 1)
              {
                    constante_curva();
                    sp_velocidade_md = (sp_velocidade_md_new * constante_lo_new);
                    sp_velocidade_md_new = sp_velocidade_md;
                    sp_velocidade_me = (sp_velocidade_me_new * constante_hi_new);
                    sp_velocidade_me_new = sp_velocidade_me;
                    calcula_pwm(); // calcula o pwm  de cada roda
                    aciona_motor(); // aciona os motores
              }else{
                    sp_velocidade_md = (sp_velocidade_md_new * constante_lo);
                    sp_velocidade_md_new = sp_velocidade_md;
                    sp_velocidade_me = (sp_velocidade_me_new * constante_hi);
                    sp_velocidade_me_new = sp_velocidade_me;
                   }
            }
          if(sensor == 2) //Sensor da esqueda na linha
            {
              if (cont_loop > 1)
              {
                    constante_curva();
                    sp_velocidade_me = (sp_velocidade_me_new * constante_lo_new);
```

```
                    sp_velocidade_me_new = sp_velocidade_me;
                    sp_velocidade_md = (sp_velocidade_md_new * constante_hi_new);
                    sp_velocidade_md_new = sp_velocidade_md;
                    calcula_pwm(); // calcula o pwm  de cada roda
                    aciona_motor(); // aciona os motores
                  }else{
                    sp_velocidade_me = (sp_velocidade_me_new * constante_lo);
                    sp_velocidade_me_new = sp_velocidade_me;
                    sp_velocidade_md = (sp_velocidade_md_new * constante_hi);
                    sp_velocidade_md_new = sp_velocidade_md;
                    }
                }
               if(sensor == 0)
                {
                  cont_loop = 0;
                  sp_velocidade_md_new = sp_velocidade;
                  sp_velocidade_md = sp_velocidade;
                  sp_velocidade_me_new = sp_velocidade;
                  sp_velocidade_me = sp_velocidade;
                }
}//FIM seta_direcao
int constante_curva()
{
   constante_hi_new = (constante_hi*1.01);
   constante_lo_new = (constante_lo*0.99);
}//FIM constante_curva
stop()
{
   pwm_md = 0;
   pwm_me = 0;
   aciona_motor(); // aciona os motores
}//FIM Stop()
int le_sensor_direcao()
{
int sensor_ir_e;
int sensor_ir_d;
   sensor_ir_e = !PORTC.RC4;//entrada pino 23 invertida.
   sensor_ir_d = !PORTC.RC5; //entrada pino 24 invertida.
   if((sensor_ir_d == 1) || (sensor_ir_e == 1))// testa se algum dos sensores esta na linha
     {
       if((sensor_ir_d == 1) && (sensor_ir_e == 1))// se os dois sensores IR leram a linha fim de curso.
         {
           stop();
           Lcd_Custom_Cmd(LCD_CLEAR);
           Lcd_Custom_Out(1, 1, "Fim da linha");// Escreve no display (linha, coluna).
           Lcd_Custom_Out(2, 1, "START -> INICI0");// Escreve no display (linha, coluna).
           while (!start_button());// aguarda a tecla START ser pressionada
         }
       if((sensor_ir_d == 1) && (sensor_ir_e == 0))// se o sensor IR da direita leu a fita
         {
           sensor = 1;
          cont_loop++;
         }
       if((sensor_ir_d == 0) && (sensor_ir_e == 1))// se o sensor IR da esquerda leu a fita
         {
           sensor = 2;
           cont_loop++;
         }
     }else
         {// se nenhum dos sensores leu a fita
           sensor = 0;
           cont_loop = 0;
         }
    return sensor;
}//FIM le_sensor_direcao
int le_velocide()
```

```
{
char var_rc0;
char var_rc1;
     if (PORTC.RC7 == 1)// DIP 4 lê se segue a linha
     {
       segue_linha = 1;
       }
     var_rc0 = PORTC.RC0; // DIP 1
     var_rc1 = PORTC.RC3; // DIP 2
     if ((var_rc0 + var_rc1) == 0)
     {
        sp_velocidade = 4.3; // 4.3
     }
     if ((var_rc0 == 1)&&(var_rc1 == 0))
     {
        sp_velocidade = 6.0;//6.0 m/min
     }
     if ((var_rc0 == 0)&&(var_rc1 == 1))
     {
        sp_velocidade = 8.0;//8.0 m/min
     }
     if ((var_rc0 + var_rc1) == 2)
     {
        sp_velocidade = 10.0; //10.0 m/min
     }
     sp_velocidade_me = sp_velocidade;
     sp_velocidade_md = sp_velocidade;
     sp_velocidade_me_new = sp_velocidade;
     sp_velocidade_md_new = sp_velocidade;
} //FIM le_velocide
int calcula_pwm()
{
float kp_md, kp_me;
   kp = 0.6;
   ki = 0.3;
   erro_md = (sp_velocidade_md - velocidade_md);
   kp_md = (erro_md*kp);
   ki_md = (ki*ki_term_md);
   pid_md = (kp_md + ki_md);
   if((pid_md > 10.67)||(pid_md < 2.365))
     {
       if(pid_md > 10.67)
         {
            pwm_md = 100;
         }else{
                pwm_md = 35;
              }
     }
     else{
           pwm_md =((7.82*pid_md)+16.5);
           ki_term_md += erro_md;
         }
      erro_me = (sp_velocidade_me - velocidade_me);
      kp_me = (erro_me*kp);
      ki_me = (ki*ki_term_me);
      pid_me = (kp_me + ki_me);
      if((pid_me > 10.67)||(pid_me < 2.365))
        {
          if(pid_me > 10.67)
            {
               pwm_me = 100;
            }else{
                   pwm_me = 35;
                 }
        }
        else{
```

```
            pwm_me =((7.82*pid_me)+16.5);
            ki_term_me += erro_me;
          }
} //FIM calcula_pwm;
int le_encoder_esq()// função que lê a medida da largura do pulso do encoder do motor da esquerda
{
int z;
unsigned long soma_rb4 = 0;
   encoder = 1;// seta 0 pra ler o encoder do motor da esquerda que esta no pinos RB4
   TRISB.RB5  = 0;
   TRISB.RB4  = 1;
   INTCON.RBIE  = 1; // desabilita o interrupção
   while (var_vet_rb4 < 4){}// aguarda até que tenha tido 4 amostras
   for (z=2;z<4;z++)// soma as amostras de 2 a 4 = a 3 amostras
   {
     soma_rb4 = soma_rb4 + t_rb4_vet[z];
   }
   t_me_med  = (soma_rb4/3);// faz a media do tempo dividindo por 8 para cada pulso
   INTCON.RBIE  = 0;// desabilita o interrupção
   var_vet_rb4 = 1;
   TRISB.RB4  = 0;
   velocidade_me = ((1354891/89849)/(t_me_med/3125));
}// FIM le_encoder_esq
int le_encoder_dir()// função que lê a medida da largura do pulso do encoder do motor da direita
{
int d;
unsigned long soma_rb5 = 0;
   encoder = 2;// seta 0 pra ler o encoder do motor da direita que esta no pinos RB5
   TRISB.RB4  = 0;
   TRISB.RB5  = 1;
   INTCON.RBIE  = 1;// habilita o interrupção
   while (var_vet_rb5 < 4){}// aguarda até que tenha tido 4 amostras
   for (d=2;d<4;d++)// soma as amostras de 2 a 4 = a 3 amostras
   {
     soma_rb5 = soma_rb5 + t_rb5_vet[d];
   }
   t_md_med = (soma_rb5/3);// faz a media, dividindo por 3
   INTCON.RBIE  = 0;// desabilita o interrupção
   var_vet_rb5 = 1;
   TRISB.RB5  = 0;
   compensador = ((pwm_md - 49.42)/181.94);
   velocidade_md = (((1354891/89849)/(t_md_med/3125)) - compensador);
   return velocidade_md; // retorna a media do tempo em que o sinal ficou em nível alto
}// FIM le_encoder_dir
int ini_pwm()//configura os pwms
{
   Pwm1_Init(5000);// Inicializa PWM1 pino 17 (RC2/CCP1/P1A)
   Pwm1_Start();// Start PWM
   Pwm2_Init(5000);// Inicializa PWM1 pino 16 (RC1/T1OSI/CCP2(1))
   Pwm2_Start();// Start PWM
}
start_button()// Le a tecla Start
{
int start;
   start = PORTD.RD3;
return start;
}//FIM start_button
int setup_io()// configuras as portas como entrada ou saída
{
  ADCON1 = 0b00101111;// configura as analógicas como digital
  TRISB  = 0b00000000;// habilita os Bits de entrada
  TRISC  = 0b11111001;// habilita os Bits de entrada
  return 0;
}// FIM setup_io
int ini_interrupt()//configura as interrupções
{
```

```
  RCON.IPEN = 0;
  INTCON  = 0b11000000;// habilita para configurar todas as interrupções
  T0CON   = 0b00000010;// habilita o Prescaler do TIMER0  pra 8
  T1CON   = 0b00110001;// habilita o Timer1 com prescaler de 8. um ciclo a cada 1,6micro segundos
  PIR1.TMR1IF = 0;// clear TMR1IF
  return 0;
}// FIM ini_interrupt
int ini_lcd()// inicialização do display
{
  Lcd_Custom_Config(&PORTD,7,6,5,4,&PORTD,0,1,2);// Configuração dos pinos do LCD
  Lcd_Custom_Cmd(LCD_CURSOR_OFF);// Desabilita o cursor piscando
  Lcd_Custom_Cmd(LCD_CLEAR); // Limpa display
  return 0;
}//FIM ini_lcd
int aciona_motor()// rotina para acionar os motores
{
  Pwm1_Change_Duty((pwm_md * 255)/100);
  Pwm2_Change_Duty((pwm_me * 255)/100);
}//FIM aciona_motor
void interrupt(void)// função que trata as interrupções.
{
   if(INTCON.RBIF == 1)//trata o interrupção a nos pinos RB4 a 7
   {
   INTCON.RBIF = 0;
    if (encoder == 2)
    {
      if (PORTB.RB5 == 1)// encoder da esquerda
      {
         TMR0L = 0;
         TMR0H = 0;
         T0CON.TMR0ON = 1;
      }
      else
      {
         T0CON.TMR0ON = 0;
         timer0L = TMR0L;// carrega a parte baixa
         timer0H = TMR0H;// carrega a parte alta
         TMR0L = 0;
         TMR0H = 0;
         t_rb5_vet[var_vet_rb5] = ((timer0H << 8) + timer0L);// soma parte alta e a baixa e grava no
vetor
         var_vet_rb5++;
       }
    }
    if (encoder == 1)   // digo qual dos RB ele vai ler
    {
     if (PORTB.RB4 == 1)//verifica se é de borda de subida
     {
       TMR0L = 0;
       TMR0H = 0;
       T0CON.TMR0ON = 1;
     }
     else
     {
       T0CON.TMR0ON = 0;
       timer0L = TMR0L;// carrega a parte baixa
       timer0H = TMR0H;// carrega a parte alta
       TMR0L = 0;
       TMR0H = 0;
       t_rb4_vet[var_vet_rb4]= ((timer0H << 8) + timer0L);// soma parte alta com a baixa e grava no
vetor
       var_vet_rb4++;
     }
    }
  }
  if(PIR1.TMR1IF == 1)// gera a interrupção a cada 104,856ms
```

```
    {
      PIR1.TMR1IF = 0;// clear TMR1IF
      TMR1H = 0x00;
      TMR1L = 0x00;
      cont_timer++;
    }// FIM da interrupção TMR1IF
}// FIM da INTEPURUPT
```