## 

Engenharia de Controle e Automação

## GNE386 - Laboratório Integrador

## 

## **Projeto Desafio: Robô Seguidor de Linha com Controle Analógico**

## 

## 

Docente:

Prof. Danilo Alves de Lima

Discentes:

Luan Junior R S Lagoas

Mateus Rodrigues Santos

Turma 22B

## 

Lavras - MG

Jul/2018

## Robô Seguidor de Linha com Controle Analógico

1. Introdução

A criação de robôs, e de todo o escopo da robótica, é oriunda das transformações causadas na revolução industrial que permitiram o desenvolvimento de máquinas cada vez mais capazes de realizar e reproduzir tarefas humanas de forma autônoma ou não. Muito além de simplesmente substituir tarefas humanas repetitivas, as máquinas desenvolvidas chegam além, realizando atividades fora da capacidade humana, seja braçal, mental, ou até mesmo substituindo os homens em serviços de alta periculosidade [1].

Hodiernamente, a expansão do termo “robô”, proporcional à expansão de aplicabilidade das máquinas tornou a indústria da automação e controle um patamar básico da engenharia. O termo “controle” diz respeito à parcela de desenvolvimento nesse escopo onde é estudada e implementada a teoria de controle. Que significa o tratamento matemático do comportamento do sistema dinâmico para que uma determinada variável de saída do sistema obedeça um ponto de referência desejado para o cumprimento do proposto [2].

Como projeto desafio, tomado por avaliação final da disciplina de Laboratório Integrador (GNE386), do curso de Engenharia de Controle e Automação, propõe-se a implementação de um sistema de controle em uma planta que se diferencia daquelas já estudadas ao longo da disciplina. Dessa forma, propõe-se a construção de um robô seguidor de linha, com controle totalmente analógico.

Robôs seguidores de linha são amplamente utilizados em escopo didático para o ensino da robótica, além de sua utilização para a promoção de competições de desenvolvimento de tecnologia. Seu objetivo consiste em, autonomamente, seguir um trajeto especificado por uma linha desenhada no chão, de forma mais veloz possível, sem perder a referência da linha em nenhum momento [3]. Neste trabalho, foi feito o projeto e implementação de um robô móvel que cumprirá esse objetivo sem o uso de microcontroladores programáveis, ou seja, toda a implementação do sistema de controle foi projetada para a atuação a partir de circuitos analógicos, que serão explicados a fundo neste documento. O documento se divide em cinco seções, sendo este introdutório, os objetivos do trabalho, materiais e métodos utilizados no projeto, modelagem do sistema dinâmico, e, por fim, o projeto e resultados obtidos com o controlador analógico.

2. Objetivos

Este projeto tem como objetivo a aplicação dos conhecimentos adquiridos ao longo do curso de Engenharia de Controle e Automação, principalmente da disciplina de Laboratório Integrador, na execução de um projeto real de sistema de controle, que atenda aos requisitos solicitados. Para cumprimento do proposto, serão seguidos os passos:

* Projeto do Hardware a ser implementado;
* Construção do robô;
* Modelagem do sistema dinâmico;
* Validação do modelo obtido no sistema construído a partir da observação das saídas face a determinadas entradas;
* Sintonia do controlador para dimensionamento dos parâmetros iniciais;
* Testes do sistema de controle para otimização dos parâmetros.

3. Materiais e Métodos

3.1. Materiais

A planta será composta por um driver com dois choppers de classe E (ponte H), utilizando-se o componente L298 [4]; dois motores DC de 9 volts acoplados às rodas que movimentarão o robô [5]; 8 sensores infravermelhos para detecção da cor da pista, o sensor utilizado será o TCRT5000 [6]; um conversor digital/analógico para transformação do sinal dos oito sensores em um só sinal analógico que indicará a posição do robô em relação à linha traçada. O DAC utilizado será o DAC0800 [7]; e por fim o compensador proporcional integral e derivativo (PID), assim como todas as outras manipulações matemáticas do sistema será implementado a partir de amplificadores operacionais. Os Amplificadores utilizados serão da classe lm741 [8].

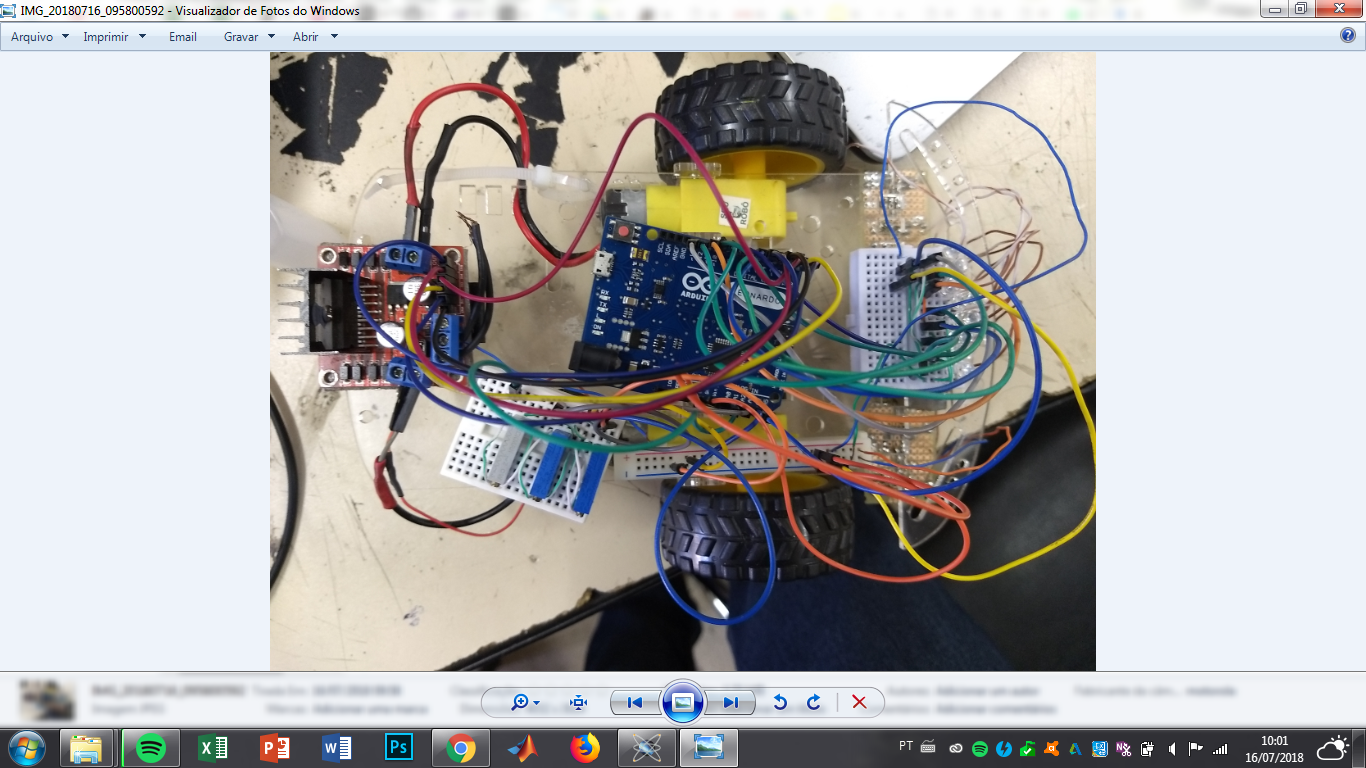


Figura 1 - Montagem eletrônica do robô completo. [Fonte própria]

3.2. Projeto Eletrônico

A seguir, serão apresentados os parâmetros de projeto, juntamente ao esquemático eletrônico das partes que compõem o circuito do robô.

3.2.1. Sensores

Os sensores reflexivos ópticos possuem, no mesmo dispositivo, um LED infravermelho como emissor de sinal luminoso, e um fotodiodo de sensibilidade no espectro infravermelho como receptor do sinal. A Figura 2 demonstra sua construção física. Sua atuação é baseada na percepção do reflexo do sinal emitido pelo LED. Sua aplicação é bastante ampla, sendo utilizado na detecção de pequenas distâncias, e principalmente na identificação de superfícies. No escopo dessa aplicação, utilizou-se o sensor de forma a emitir um sinal lógico baixo (próximo a 0V) ao identificar uma superfície de cor preta, e ocorre a emissão de sinal lógico alto (próximo à 5V) face à identificação de superfície branca.



Figura 2 - Sensor Óptico Reflexivo TCRT5000 [6].

Para o funcionamento conforme descrito, o sensor fora inserido no circuito conforme demonstra a Figura 3, onde os sinais de saída serão explicados mais à frente neste documento.

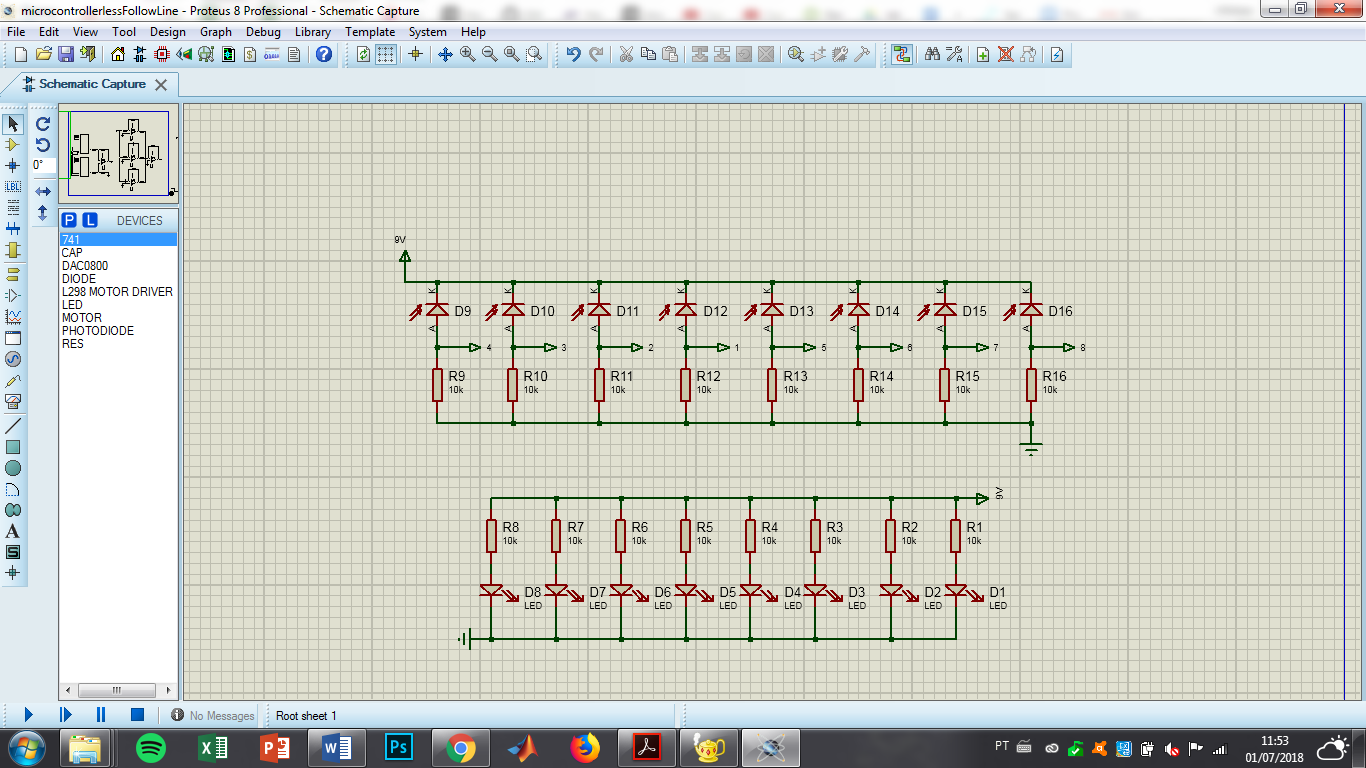


Figura 3 - Esquemático eletrônico da malha de sensoriamento do robô. [Fonte própria]

3.2.2. Conversor Digital Analógico e Sinal de Referência

Conforme descrito no tópico anterior, cada sensor atua emitindo sinais lógicos alto ou baixo, de acordo com a cor percebida na superfície. Para a explicação do circuito de geração do sinal de erro a partir desse sensoriamento é importante remeter à configuração de montagem de robô. São montados 8 sensores enfileirados apontando para a superfície abaixo do robô. Seu estado de referência consiste nos dois sensores centrais em nível lógico alto, e os demais em nível lógico baixo, formando a palavra binária “00011000”. A alteração nos bits marginais da palavra binária indica que a linha identificada está saindo da posição central do robô, dessa forma deve ser gerado um sinal de erro para que os motores possam corrigir a posição.

Entretanto, foi visto um problema nessa configuração, uma vez que ela gerava sinais de erro desproporcionais para curvas à esquerda e curvas à direita. Por exemplo, se fosse lida a palavras binária “01110000” o sinal de erro seria muito maior que a leitura de uma curva para o lado contrário “00001110”, devido à esses sensores corresponderem à bits menos significativos na palavra. A solução encontrada foi a divisão dos sensores em duas palavras de 4 bits, onde os primeiros 4 sensores correspondem ao sinal da esquerda do robô, e os últimos 4 ao sinal da direita. Utilizando-se dois conversores DA, o sinal foi tratado como duas palavras binárias distintas, onde o estado de referência corresponde às duas palavras em “0001”. Os sinais analógicos de saída dos conversores são subtraídos utilizando-se um amplificador subtrator, que gera o sinal de erro. se a subtração dos sinais for igual a 0, o sistema não tem erro, erros negativos indicam que o sistema precisa corrigir a posição em uma direção, e positivo, em outra.

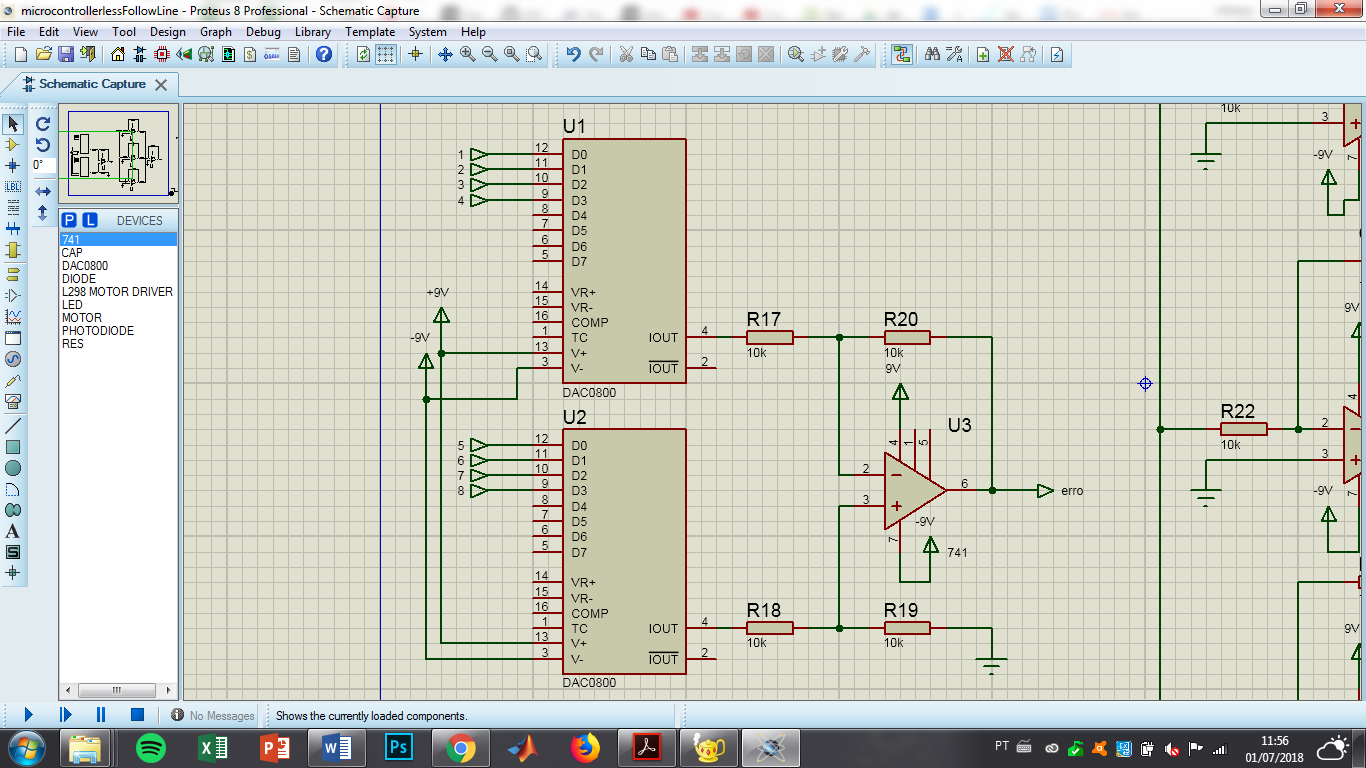


Figura 4 - Esquemático eletrônico DAC e Subtrator. [Fonte própria]

3.2.3. Controlador PID

Com o tratamento dos sensores feito conforme descrito no item anterior, obtém-se um sinal correspondente ao erro entre o estado de referência de posição do robô (aquele desejado) e o estado real lido pelos sensores. Objetiva-se utilizar este sinal para o controle da variável posição, fazendo com que este erro indique ao driver controlador dos motores qual atuação deve ser tomada para que o robô retorne ao estado de referência. A transformação do sinal de erro em sinal de controle é feita pelo tratamento matemático desse erro por uma função conhecida como controlador.

Os controladores cumprem o objetivo de influenciar o comportamento dinâmico do sistema, controlando a resposta de saída (tensão nos motores) de acordo com o comportamento do sinal de erro ao longo do tempo [2]. A técnica de controle a ser utilizada é o Controle PID (Proporcional, Integral, Derivativo), que utiliza essas três técnicas matemáticas para minimizar o sinal de erro, zerar o sinal de erro, e diminuir sua oscilação, respectivamente [9][10].

A implementação eletrônica do controlador PID está representada na Figura 5. Onde U4 representa o amplificador proporcional, U5 o amplificador integral, U6 o amplificador derivativo, e por fim U7 é um amplificador somador que sintetiza os três sinais.

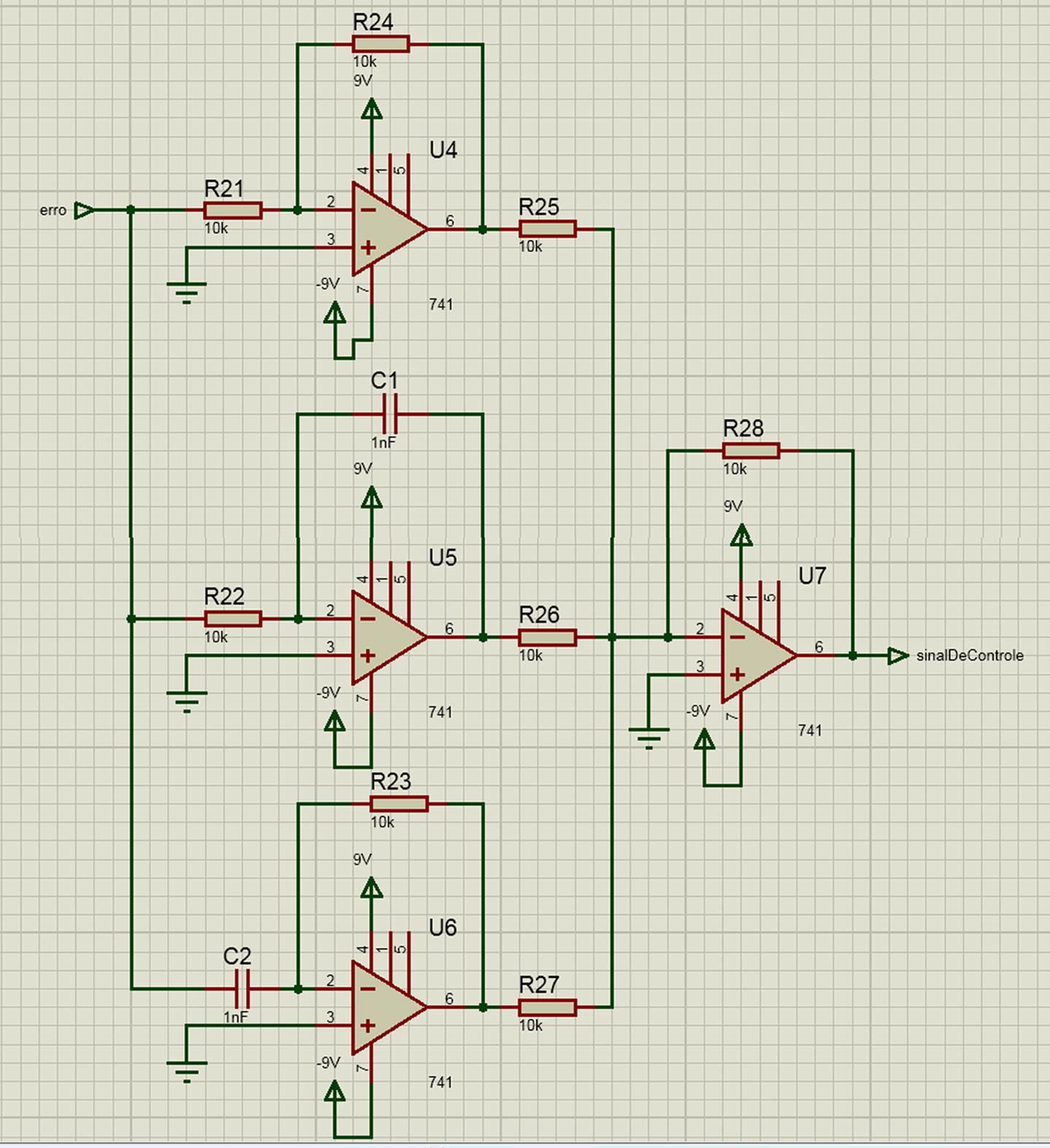
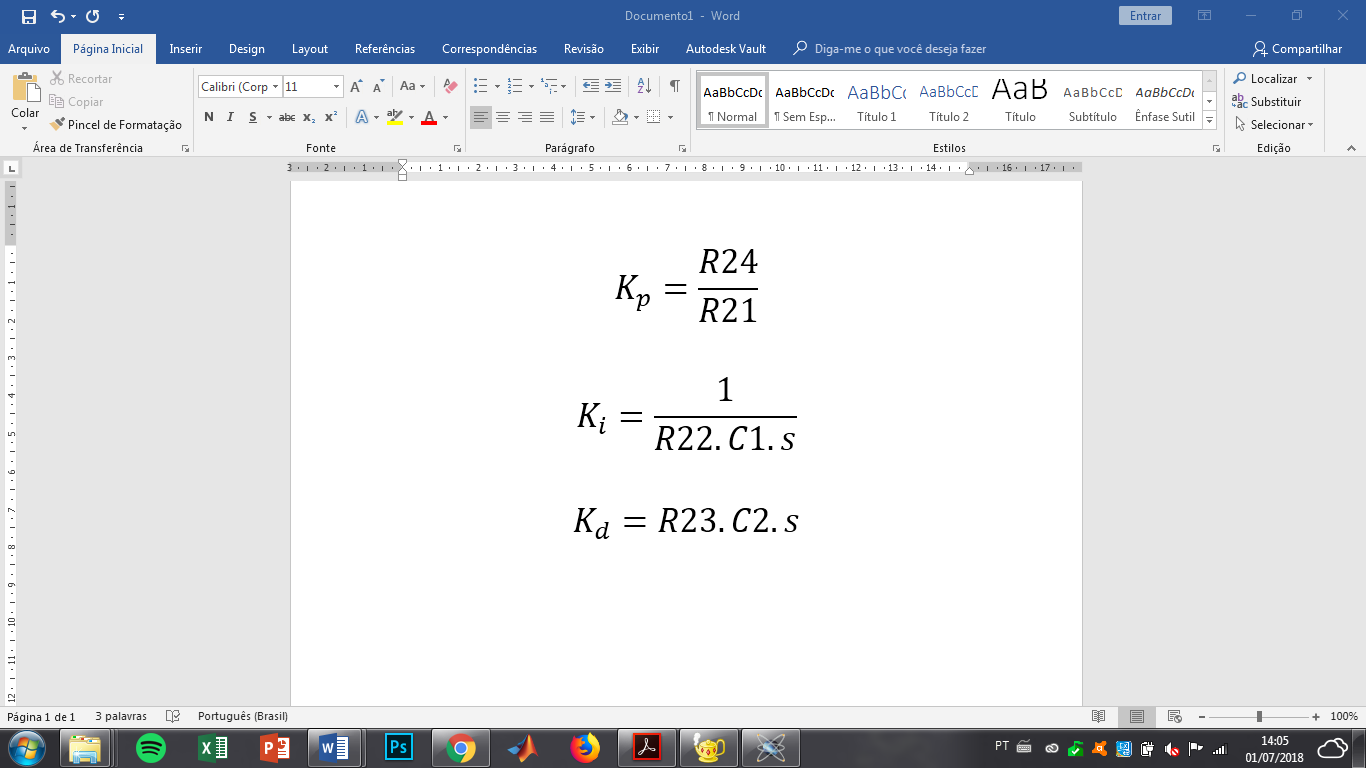


Figura 5 - Esquemático eletrônico do controlador PID. [Fonte própria]

A definição dos parâmetros do controlador PID é feita por meio dos resistores R21, R22, R23, e R24, além dos capacitores C1 e C2. Mais adiante no documento será abordada a modelagem matemática do sistema, onde os parâmetros serão contextualizados, porém, os ganhos do controlador, Kp, Ki e Kd, podem ser calculados no circuito como [11]:



no domínio s. Para a alteração dos parâmetros de acordo com os testes executados, os resistores R21, R22 e R23 são montados como *trimpots*, componentes de resistência variável, manipulada manualmente.

3.2.4. Tratamento do Sinal de Controle e Driver L298

O sinal de controle, assim como o sinal de erro, consistirá em um sinal positivo ou negativo conforme posicionamento a ser corrigido. Entendendo o comportamento do driver, composto por duas pontes H para controle distinto de dois motores, o driver é composto de 4 entradas de sinal (além de 3 entradas de alimentação, e 4 saídas, 2 para cada motor). Dentre as 4 entradas de sinal, 2 representam o controle do motor A, e as outras duas do motor B. O comportamento de cada motor é dado pela combinação dos sinais em cada entrada [4].

No caso deste trabalho, deseja-se que o robô se controle andando apenas para frente sempre. Sendo assim, as entradas do driver In2 e In3 foram curto circuitadas e aterradas, e o sinal de controle é aplicado somente nas entradas In1 e In4, fazendo com que os motores funcionem de forma espelhada e o controle atue com o robô andando para frente em intensidades variáveis na rotação de cada roda, permitindo que faça as curvas necessárias.

A definição do sinal aplicado em cada motor é dada pelo circuito retificador que aplica sinais de controle positivos ao motor A, e negativos ao motor B, fazendo com que cada um desses altere sua rotação proporcionalmente ao controle. A Figura 6 ilustra o circuito descrito.

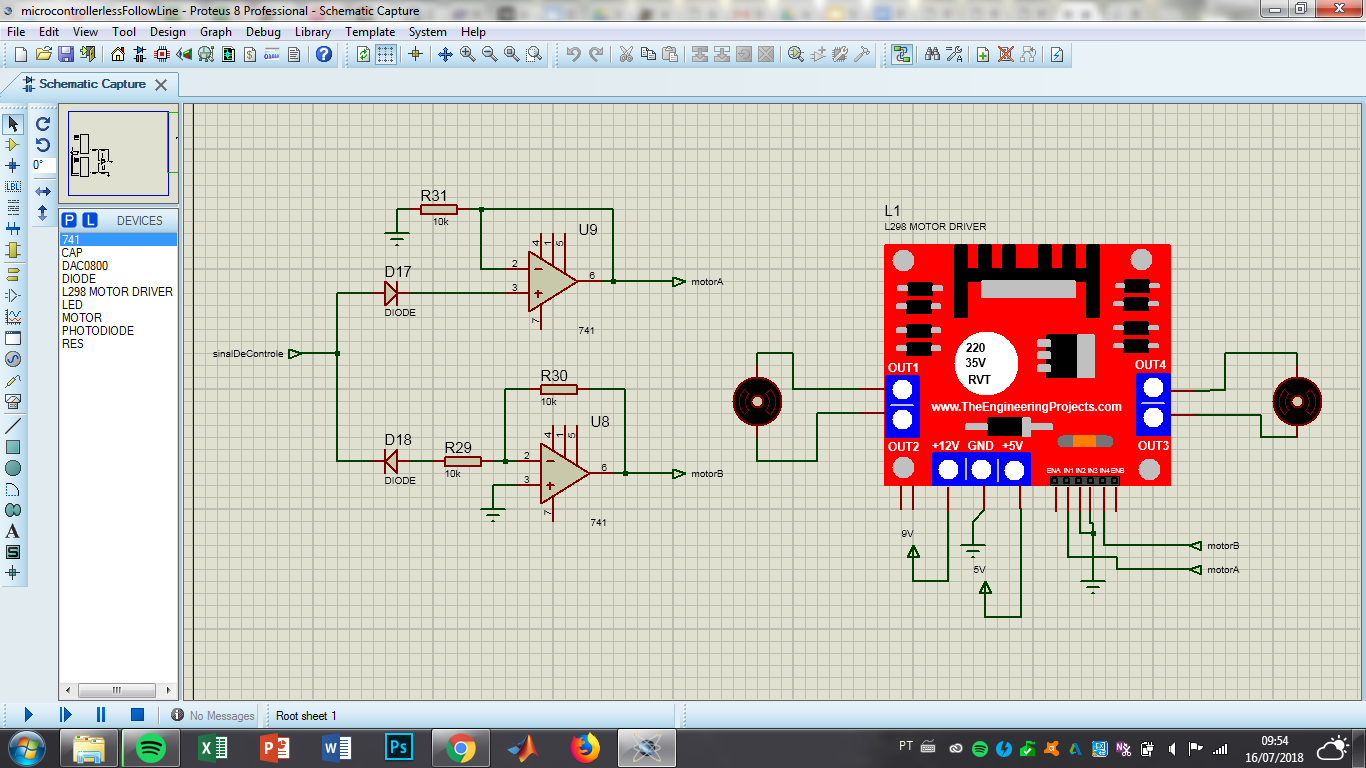


Figura 6 - Esquemático eletrônico Retificador e Driver L298. [Fonte própria]

4. Modelagem

Foi feito um ensaio para medir as velocidades do robô variando o duty cicle. Os dados obtidos foram linearizados, obtendo a Equação 1, que relaciona a velocidade em m/s com o duty cicle do PWM usado para acionar os motores.

(1)

Os resultados obtido no ensaio estão apresentados na Figura 1.

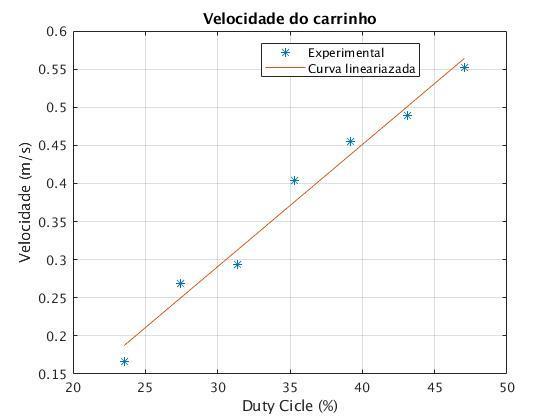


Figura 7: Velocidade do robô em função do duty cicle.

Considerou-se no projeto uma velocidade base de 31,4%. A partir disso, definiu-se as velocidades dos motores por meio das Equações 2 e 3,

(2)

(3)

Mediu-se a distância da placa de sensores ao eixo das rodas. O valor obtido foi de 6 cm.

Medindo-se as distância entre os sensores e o centro de simetria do robô, foi possível calcular o raio da curva necessária para corrigir usando o teorema de pitágoras. Os resultados encontrados estão apresentados na Tabela 1.

Tabela 1: Raio da curvatura necessária para corrigir o erro detectado pelos sensores.

|  |  |  |
| --- | --- | --- |
| Sensor | Distância ao centro (m) | Raio da curva (m) |
| esq3 | 0.0595 | 0.0600 |
| esq2 | 0.0405 | 0.0647 |
| esq1 | 0.0215 | 0.0945 |
| esq0 | 0.0025 | 0.7212 |
| dir0 | 0.0025 | 0.7212 |
| dir1 | 0.0215 | 0.0945 |
| dir2 | 0.0405 | 0.0647 |
| dir3 | 0.0595 | 0.0600 |

Mediu-se a distância entre as duas rodas. O valor obtido foi de 11.5 cm.

Considerando a velocidade base com duty cicle de 31,4%, estimou-se o controle necessário para fazer o robô desenvolver uma trajetória circular com variados raios. Os valores obtidos estão apresentados na Tabela 2.

Tabela 2: Raio do círculo para uma valor de controle constante.

|  |  |
| --- | --- |
| Raio (m) | Controle (%) |
| 0.05 | 38.1402 |
| 0.1 | 25.2507 |
| 0.2 | 18.8059 |
| 0.4 | 15.5836 |
| 0.6 | 14.5095 |
| 0.8 | 13.9724 |
| 1 | 13.6502 |

5. Controle

Conforme supracitado, o mal funcionamento do circuito analógico projetado acarretou na mudança de concepção do projeto para a utilização de um controlador digital, no caso, fora utilizada a plataforma de desenvolvimento *open source* Arduino.

A mudança de concepção do projeto alterou a necessidade de controle para um controle em tempo discreto. O compensador escolhido para a planta foi o Proporcional Integral Derivativo (PID), sua escolha e funcionamento serão abordadas a frente. O Arduino, neste projeto, cumpriu o papel de todos os circuitos analógicos intermediários entre a placa de sensoriamento e o driver de potência, que continuaram os mesmos conforme descrito na seção anterior.

O código completo implementado no microcontrolador pode ser encontrado no Anexo 1 deste documento.

5.1. Implementação e Análise do Controlador PID

O tipo de controlador mais amplamente estudado e utilizado para fins didáticos ou não. Visto em todas as plantas no escopo da disciplina, este controlador trata o sinal de erro a partir de três parâmetros, fazendo o ajuste para seu comportamento dinâmico de forma a eliminar inconvenientes do controlador básico (On-Off).

Seu modelo no sistema já fora descrito na seção 3.2.3, e sua implementação em sistema discretizado, no Arduino, é feita da forma a seguir:

erro = dirbit3\*8 + dirbit2\*4 + dirbit1\*2 + dirbit0

- esqbit3\*8 - esqbit2\*4 - esqbit1\*2 - esqbit0;

integral += erro\*smptime;

derivada = (erro-lasterro)/smptime;

controle = (D\*derivada + I\*integral + P\*erro);

lasterro = erro;

Os testes no controle e análise empírica de seu funcionamento proporcionou algumas considerações, listadas a seguir.

* Pode-se perceber com clareza a ação proporcional no tratamento do erro, amplificando a ação do atuador de acordo com o erro obtido nos sensores. Para este controlador foi percebido que valores abaixo de 80 fizeram com que o robô não respondesse às curvas, a ação de controle ficou fraca e, sendo assim a linha era perdida previamente à correção. Valores entre 80 e 150 faziam com o que robô fosse capaz de acompanhar o circuito, e valores maiores que 150 propiciavam uma correção extremamente brusca nas curvas, fazendo com que o robô entrasse em zigzag e perdesse a trilha. Em ajuste fino final, o valor de ganho proporcional Kp foi ajustado em 115.
* A ação integral do controlador, introduzida principalmente para amenização do erro de offset. Entretanto por meio dos testes foi percebido que a ação integral era desnecessária no sistema, e este ganho foi, então configurado para 0.
* Por fim, foi analisada a ação derivativa, que cumpre o papel de atuação antecipada, evitando que o desvio se torne maior, ou seja, controlando o tempo de resposta das oscilações no erro. A configuração do ganho derivativo também foi feita de forma empírica, e foi descoberto que valores muito pequenos vinham causando nenhuma interferência na suavização das curvas. Valores muito altos causavam oscilações bruscas que atrapalham o controle. A suavização perfeita das curvas foi encontrada com a definição deste ganho em Kd = 3,8.

5.2. Diagrama de Blocos do Sistema

O sistema implementado e controlado pode ser ilustrado pelo diagrama de blocos da Figura 7.

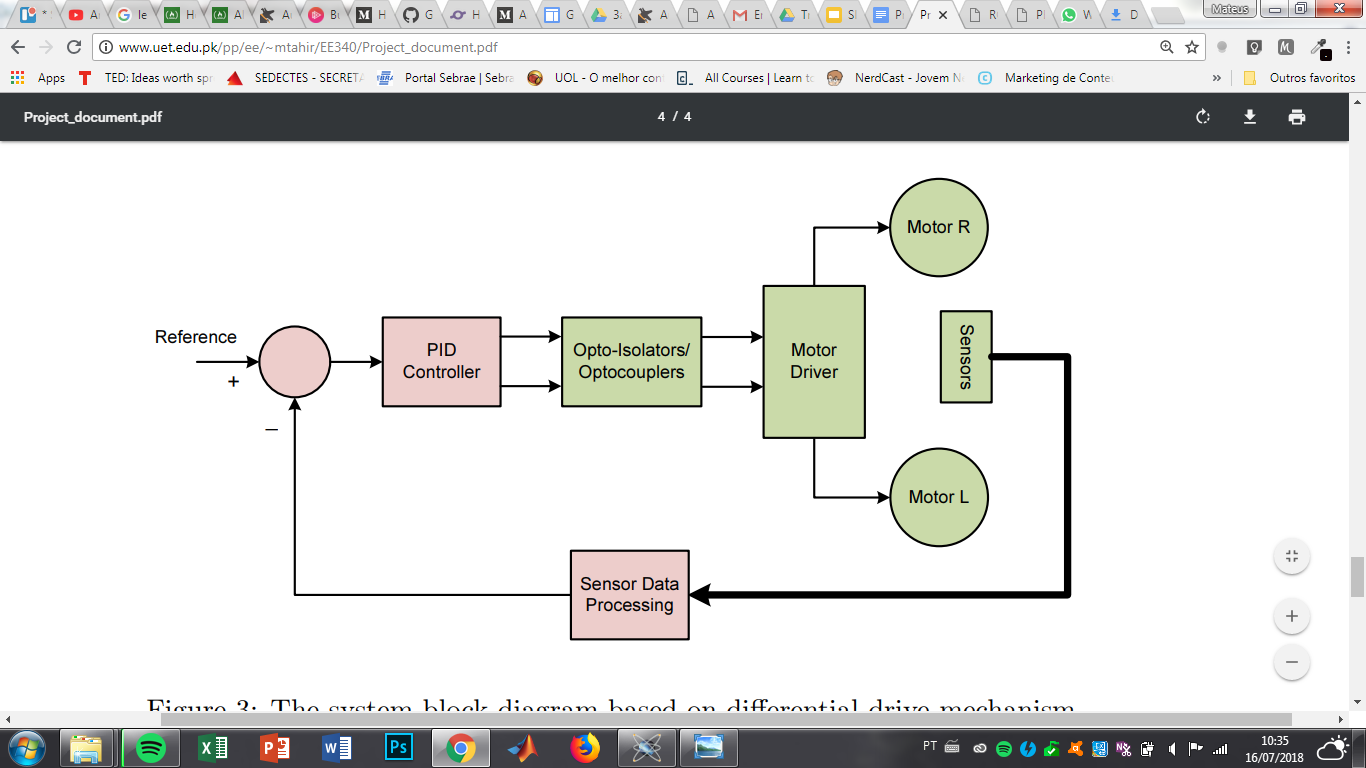


Figura 7 - Diagrama de Blocos robô seguidor de linha. [12]

6. Referência Bibliográfica

1. CITI. História da Robótica. Disponível em: <http://www.citi.pt/educacao\_final/trab\_final\_inteligencia\_artificial/historia\_da\_robotica.html>. Acesso em: 01/07/2018.
2. Wikipédia, a Enciclopédia Livre. Teoria de Controle. Disponível em: <<https://pt.wikipedia.org/wiki/Teoria_de_controle>> Acesso em: 01/07/2018.
3. Pereira, L. M. M. et. al.. Robô de Competição Categoria Seguidor de Linha Utilizando Algoritmo PID e Plataforma Arduino. Disponível em: <<http://sistemaolimpo.org/midias/uploads/230c61ca8833329f9ffc867a89566dcd.pdf>> Acesso em: 01/07/2018.
4. FilipeFlop. Motor DC com Driver Ponte H L298N. Disponível em: <<https://www.filipeflop.com/blog/motor-dc-arduino-ponte-h-l298n/>> Acesso em: 01/07/2018.
5. SeuRobo. Robô Autonomo SR1. Disponível em: <<https://seurobo.com.br/robo-autonomosr-1-o-robo-autonomo-simples-com-arduino/>> Acesso em: 05/07/2018.
6. FilipeFlop. Sensor Óptico Reflexivo Fototransistor TCRT5000. Disponível em: <<https://www.filipeflop.com/produto/sensor-optico-reflexivo-tcrt5000/>> Acesso em: 01/07/2018.
7. Texas Instruments. DAC0800 8-bit Digital to Analog Converter. Disponível em: <<http://www.ti.com/lit/ds/symlink/dac0800.pdf>> Acesso em: 01/07/2018.
8. Texas Instruments. LM741 Operational Amplifier. Disponível em: <http://www.ti.com/lit/ds/symlink/lm741.pdf>> Acesso em: 01/07/2018.
9. Citisystems. O controle PID de forma simples e descomplicada. Disponível em: <<https://www.citisystems.com.br/controle-pid/>> Acesso em: 01/07/2018.
10. Wikipédia, a Enciclopédia Livre. Controle proporcional integral derivativo. Disponível em: <<https://pt.wikipedia.org/wiki/Controlador_proporcional_integral_derivativo>> Acesso em: 01/07/2018.
11. eCircuitCenter. Op Amp PID Controller. Disponível em: <<http://www.ecircuitcenter.com/Circuits/op_pid/op_pid.htm>> Acesso em: 01/07/2018.
12. University of Engineering and Technology of Lahore. EE340: Control System Lab Project: Line Following Robot. Disponível em: <<http://www.uet.edu.pk/pp/ee/~mtahir/EE340/Project_document.pdf>> Acesso em: 12/07/2018.

Anexo 1

#define motor1p1 0

#define motor1p2 1

#define motor2p1 2

#define motor2p2 4

#define motor1pwm 3

#define motor2pwm 5

#define sensordir0 6

#define sensordir1 7

#define sensordir2 8

#define sensordir3 9

#define sensoresq0 10

#define sensoresq1 11

#define sensoresq2 12

#define sensoresq3 13

boolean dirbit0 = false, dirbit1 = false, dirbit2 = false, dirbit3 = false,

esqbit0 = false, esqbit1 = false, esqbit2 = false, esqbit3 = false;

int erro = 0, lasterro = 0; //sinal de erro

int vel1 = 0, vel2 = 0; //velocidade dos motores

double controle = 0, integral = 0, derivada; //sinal de controle, integral e derivada;

double smptime;

double P = 0, I = 0, D = 0;

ISR(TIMER1\_OVF\_vect) // Rotina responsavel pelo funcionamento do timer

{

TCNT1L = 0x7F; //Inicializa o nivel mais baixo do timer1 em 0x7F

TCNT1H = 0xC1; //Inicializa o nivel mais alto do timer 1 em 0xC1

dirbit0 = digitalRead (sensordir0);

dirbit1 = digitalRead (sensordir1);

dirbit2 = digitalRead (sensordir2);

dirbit3 = digitalRead (sensordir3);

esqbit0 = digitalRead (sensoresq0);

esqbit1 = digitalRead (sensoresq1);

esqbit2 = digitalRead (sensoresq2);

esqbit3 = digitalRead (sensoresq3);

erro = dirbit3\*8 + dirbit2\*4 + dirbit1\*2 + dirbit0

- esqbit3\*8 - esqbit2\*4 - esqbit1\*2 - esqbit0;

integral += erro\*smptime;

derivada = (erro-lasterro)/smptime;

controle = (D\*derivada + I\*integral + P\*erro);

lasterro = erro;

}

void setup()

{

pinMode (motor1p1, OUTPUT);

pinMode (motor1p2, OUTPUT);

pinMode (motor2p1, OUTPUT);

pinMode (motor2p2, OUTPUT);

pinMode (motor1pwm, OUTPUT);

pinMode (motor2pwm, OUTPUT);

pinMode (sensordir0, INPUT);

pinMode (sensordir1, INPUT);

pinMode (sensordir2, INPUT);

pinMode (sensordir3, INPUT);

pinMode (sensoresq0, INPUT);

pinMode (sensoresq1, INPUT);

pinMode (sensoresq2, INPUT);

pinMode (sensoresq3, INPUT);

TCCR1A = 0x00; //Timer 1 operando em modo normal

TCCR1B = 0x01; //Prescaler 1:1

TCNT1L = 0x7F; //Inicializa o nibel mais baixo do timer1 em 0x7F

TCNT1H = 0xC1; //Inicializa o nibel mais alto do timer 1 em 0xC1

TIMSK1 = 0x01; //Habilita interrupção do timer 1

smptime = 0.001; //Tempo de amostragem em segundos

Serial.begin (9600);

}

void loop()

{

P = analogRead (A0);

I = 0;//(double)analogRead (A1)/50;

D = (double)analogRead (A2)/50;

Serial.print (P);

Serial.print (" ");

Serial.print (D);

Serial.print (" ");

Serial.println (vel2);

vel1 = 80 - controle/10;

vel2 = 80 + controle/10;

if (vel1 > 255) vel1 = 255;

if (vel2 > 255) vel2 = 255;

if (vel1 < 40) vel1 = 0;

if (vel1 < 40) vel1 = 0;

digitalWrite (motor1p1, HIGH);

digitalWrite (motor1p2, LOW);

digitalWrite (motor2p1, HIGH);

digitalWrite (motor2p2, LOW);

analogWrite (motor1pwm, vel1);

analogWrite (motor2pwm, vel2);

}