CENTRO UNIVERSITÁRIO FIEO - UNIFIEO
ENGENHARIA DA COMPUTAÇÃO
LABORATÓRIO INTEGRADO II

Projeto Integrado: Carro Microcontrolado com Rotas Pré-Programadas

Livar da Hora
Victor de Freitas
Wagner Pinheiro
Willian Carareto
(EGC-NA8)

Osasco, Agosto de 2012

RESUMO

Com o grande avanço tecnológico nos últimos anos, a automação ganhou grande destaque no mercado, devido a sua praticidade tanto na economia, quanto na agilidade, pois era uma atividade de maior agilidade e menos força física. Seguindo a mesma linha de raciocínio, no presente projeto são detalhadas as atividades para o controle automatizado de um carro de brinquedo, utilizando para isso a plataforma Arduino como base para a programação de rotas pré-estabelecidas. Essas rotas serão controladas através do microcontrolador Atmega128 (Atmel Corp.) em conjunto de um circuito para controle dos motores contínuos. O trabalho foi desenvolvido respeitando um cronograma previsto de 14 semanas, constando nele atividades como análises, estudos, testes e execução do projeto, garantindo assim a eficiência na qualidade do hardware que faz o controle do carro, que já tem sua estrutura montada, sendo apenas adaptado com os componentes eletrônicos e mecânicos necessários para a realização do seu trajeto pré-programado sem a interferência manual direta.

SUMÁRIO

INTRODUÇÃO

O projeto a seguir mostra o desenvolvimento de um carro com uma determinada rota pré-programada, o objetivo aqui é de automatizar um carro de controle remoto.

Para esse projeto utilizaremos uma plataforma física de computação de código aberto baseado em uma placa microcontroladora chamada Arduino, essa placa pode ser usada para desenvolver objetos interativos, admitindo entradas de uma série de sensores ou chaves, e controlando uma variedade de luzes, motores ou outras saídas físicas. Projetos do Arduino podem ser independentes, ou podem se comunicar com software rodando em seu computador (como Flash, Processing, MaxMSP.). Os circuitos podem ser montados à mão ou comprados pré-montados; o software de programação de código-livre pode ser baixado de graça.

Atualmente, seu hardware é feito através de um microcontrolador Atmel AVR, sendo que este não é um requerimento formal e pode ser estendido se tanto ele quanto a ferramenta alternativa suportarem a linguagem Arduino e forem aceitas por seu projeto. Considerando esta característica, muitos projetos paralelos se inspiram em cópias modificadas com placas de expansões, e acabam recebendo seus próprios nomes.

Apesar do sistema poder ser montado pelo próprio usuário, os mantenedores possuem um serviço de venda do produto pré-montado, através deles próprios e também por distribuidores oficiais com pontos de venda mundiais.

A placa microcontroladora utilizada no projeto é a Arduino Uno, é baseada no ATmega328. Ele tem 14 pinos de entrada/saída digital (dos quais 6 podem ser usados como saídas PWM), 6 entradas analógicas, um cristal oscilador de 16MHz, uma conexão USB, uma entrada de alimentação uma conexão ICSP e um botão de reset. Ele contém todos os componentes necessários para suportar o microcontrolador, simplesmente conecte a um computador pela porta USB ou alimentar com uma fonte ou com uma bateria e tudo pronto para começar.

O Uno difere de todas as placas antecessoras no sentido de não utilizar o chip FTDI cara conversão do sinal serial. Utiliza no seu lugar um Atmega8U2 programado como conversor de USB para serial.

Cada um dos 14 pinos digitais do Uno podem ser utilizados como uma entrada ou uma saída utilizando-se as funções pinMode(), digitalWrite(), e digitalRead(). Eles operam a 5V. Cada pino pode fornecer ou receber um máximo de 40mA e tem um resistor pull-up interno (desconectado por padrão) de 20-50kΩ.

O Arduino Uno possui uma série de facilidades para se comunicar com um computador, outro Arduino, ou outros microcontroladores. O ATmega328 fornece

comunicação serial UART TTL (5V) que está disponível nos pinos digitais 0 (RX) e 1 (TX). Um ATmega8U2 na placa canaliza esta comunicação para a USB e aparece como uma porta virtual para o software no computador. O firmware do '8U2 utiliza os drivers padrão USB COM e nenhum driver externo é necessário. Entretanto, no Windows, um arquivo .inf é necessário. Ainda faltam as instruções específicas mas em breve estarão disponíveis. O software do Arduino inclui um monitor serial que permite dados textuais ser enviados e recebidos da placa. LEDs conectados ao RX e TX piscarão enquanto dados estiverem sido transmitidos pelo chip USB-para-serial e pela conexão USB (mas não para comunicação serial nos pinos 0 e 1).

Uma biblioteca de SoftwareSerial permite comunicação serial em qualquer dos pinos digitais do Uno. O ATmega328 também suporta comunicação I2C (TWI) e SPI. O software do Arduino inclui uma biblioteca Wire para simplificar o uso do bus I2C, veja a documentação para mais detalhes. Para comunicação SPI utilize a biblioteca SPI.

Utiliza-se um ambiente de desenvolvimento para escrever o código que será enviado à placa, a linguagem do Arduino é meramente um conjunto de funções C/ C++ que podem ser chamadas em seu código.

O Arduino Uno pode ser programado com o software Arduino. Simplismente selecione "Arduino Uno" no menu Tools > Board.

O ATmega328 no Arduino Uno vem pré-gravado com um bootloader que permite enviar código novo para ele sem a utilização de um programador de hardware externo. Ele se comunica utilizando o protocolo original STK500 .

O código fonte do firmware do ATmega8U2 também está disponível. Este chip é carregado com um bootloader DFU, que pode ser ativado conectando o jumper de solda na parte posterior da placa (próximo ao mapa da Itália) e depois resetando o 8U2. Você pode utilizar o software FLIP da Atmel (Windows) ou o programador DFU (Mac OS X e Linux) para carregar um novo firmware. Ou ainda utilizar um programador externo (sobrescrevendo o bootloader DFU).

Ao invés de necessitar do pressionamento físico de um botão antes de um upload, o Arduino Uno é desenvolvido que permita esta operação ser feita por meio do software rodando em um computador. Uma das linhas de controle de fluxo do hardware (DTR) do ATmega8U2 é conectado à linha de reset do ATmega328 através de um capacitor de 100nF. Quando esta linha é declarada (rebaixada) a linha de reset cai o suficiente para resetar o chip. O software do Arduino utiliza esta capacidade para permitir o envio de código novo simplismente pressionando o botão de upload na IDE. Isto significa que o bootloader pode ter um intervalo mais curto, uma vez que o rebaixamento do DTR pode ser melhor coordenado com o início do upload.

Esta configuração tem outras implicações. Quando o Uno é conectado a um computador rodando Mac OS X ou Linux, ele é resetado cada vez que uma conexão é estabelecida com o software (via USB). Durante o próximo meio segundo o bootloader estará rodando no Uno. Uma vez que ele está programado para ignorar

dados malformados (i.e. qualquer coisa diferente do upload de um novo código), ele irá interceptar os primeiros bytes de informação após a abertura da conexão. Se um programa rodando na placa recebe alguma configuração ou outra informação quando começa a rodar esteja seguro de que o software com o qual ela se comunica espere por um segundo antes de começar a enviar dados.

O Uno contem uma trilha que pode ser interrompida (cortada fisicamente) para desabilitar o auto-reset. Os conectores de cada lado da trilha podem ser soldados para reabilitar esta função. Ela está identificada como "RESET-EN".

Utilizaremos também uma carcaça de carrinho de controle remoto na qual colocaremos a placa microcontroladora, para controlar dois motores contínuos, um para virar e o outro para movimentar o carrinho à frente, alem de outros componentes eletrônicos para a automatização do mesmo.

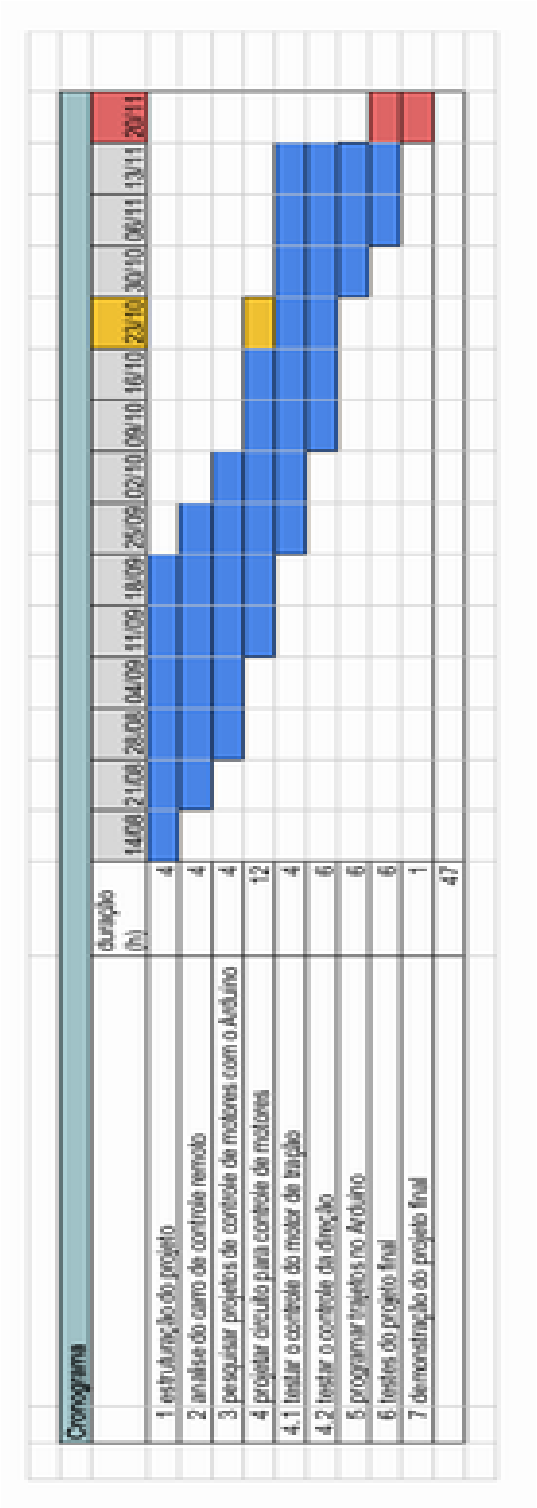
Os processos de construção e funcionamento do carro automatizado são explicados de forma clara através de imagens e documentos ao longo do projeto.

METODOLOGIA

Para o cumprimento do projeto serão consideradas as seguintes etapas:

1. estruturação do projeto
2. analise do carro de controle remoto
3. pesquisar projetos de controle de motores com o Arduino
4. projetar circuito para controle de motores
    a. compra de componentes
    b. projetar circuito
    c. criar esquema do circuito
    d. teste da API do Arduino
5. testar o controle do motor de tração
6. testar o controle da direção
7. programar trajetos no Arduino
8. testes do projeto final
9. demonstração do projeto final

CRONOGRAMA



ANALISE DO CARRO RF

O carro controlado por radio frequencia (RF) (*imagem 1*), possui um circuito impresso com componentes que controlam a recepção de sinal (40MHz) enviados por um controle remoto (*imagem 2*), alem de dois motores de corrente continua de 5V, um para controlar a tração do carro e outro para controlar a direção. Para a alimentação do circuito são utilizados 3 baterias de 1,5 volts, cada, e que ficam alojadas na parte inferior do carro.

Para controlar a tração existe um conjunto de engrenagens que transfere o movimento do motor para as rodas através de um eixo central (*imagem 3*).

Para o controle da direção existe um pequeno jogo de engrenagens que transfere a rotação do motor, de forma unica, para as rodas dianteiras (*imagem 4*).

O projeto visa substituir o circuito impresso que controla o movimento do carro, por um circuito constituído do Arduino e um controlador de motor DC acoplado ao mesmo. Para este circuito será utilizado como base o CI 293D, tendo como base o esquema da *imagem 5*.

Imagem 1: Carro Controlado por RF

Imagem 2: Visão Geral Interna do carro RF

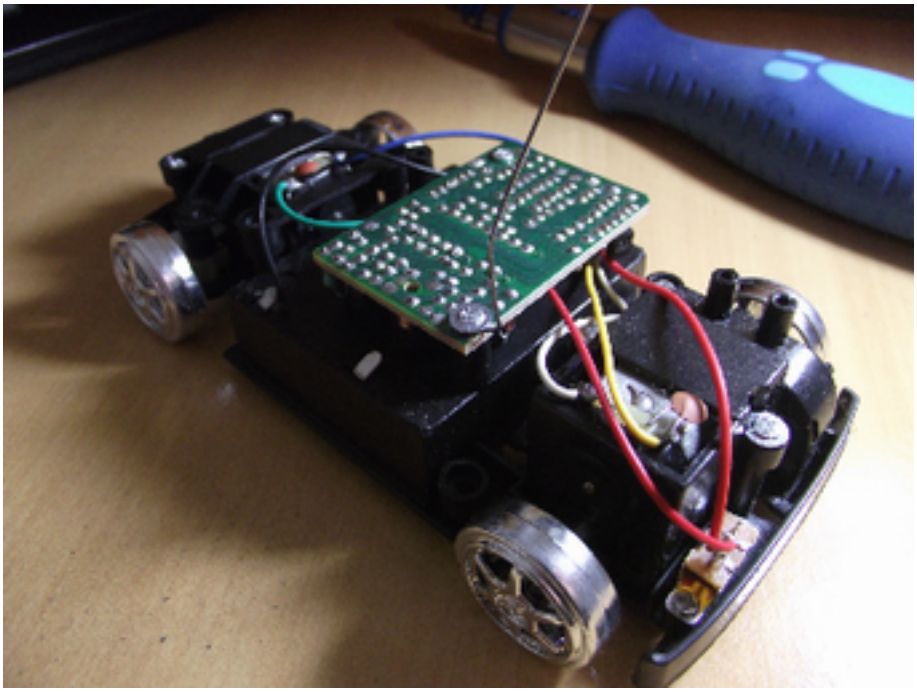


Imagem 3: Motor DC para tração do carro

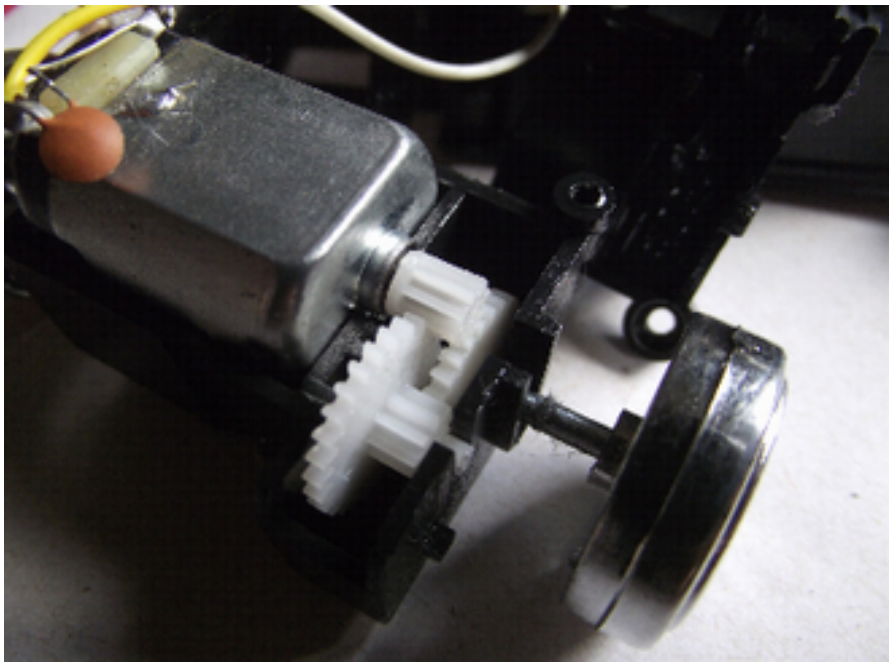


Imagem 4: Controle da direção do carro

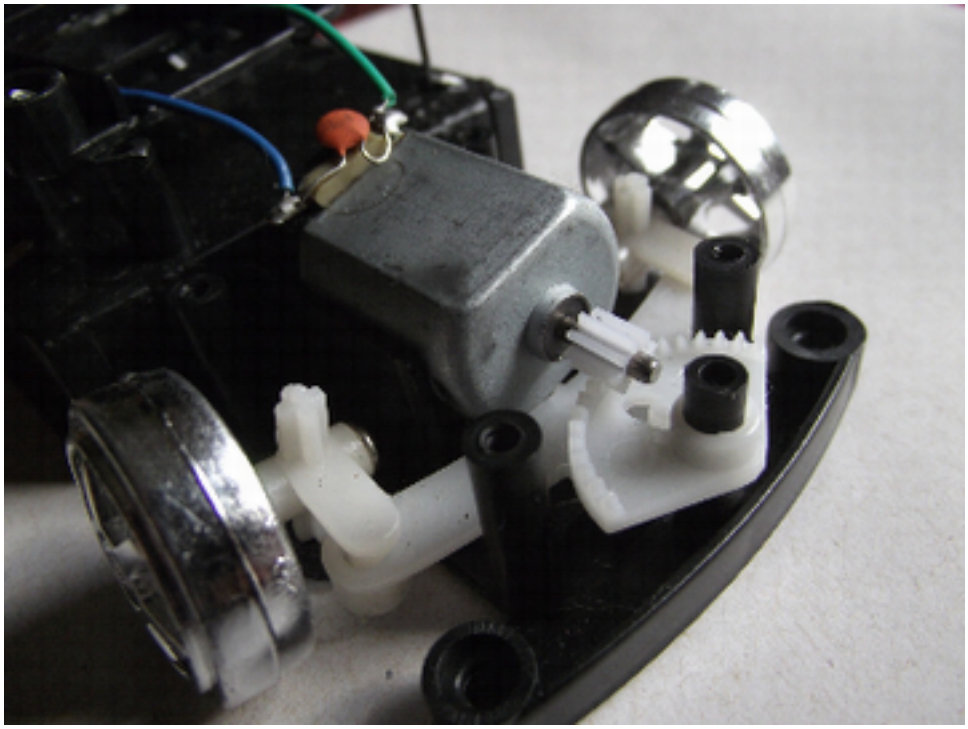
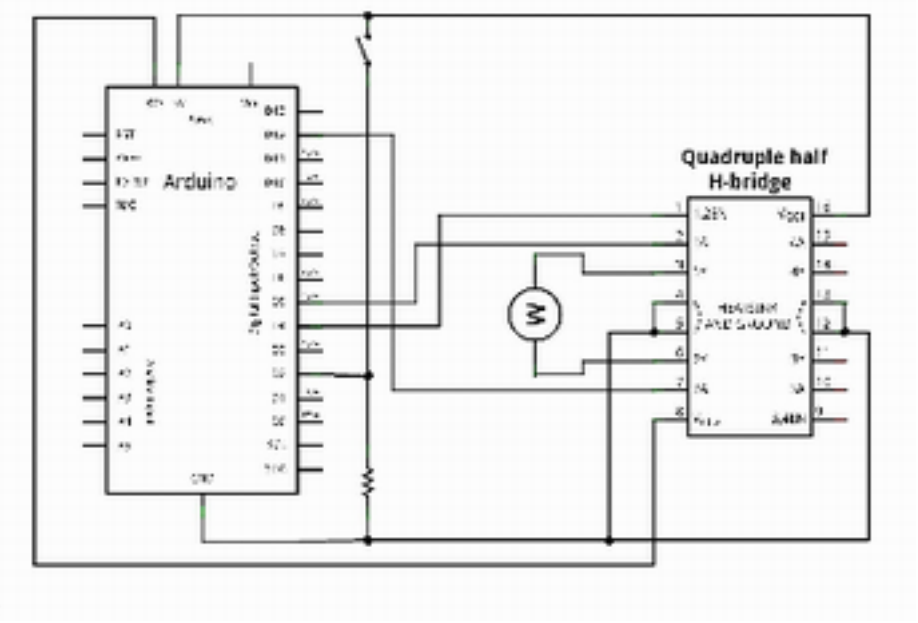


Imagem 5: Esquema modelo para o controlador dos motores



DC

PROJETANDO O CONTROLADOR DOS MOTORES DC

PONTE H

O conceito chave para o controle de motores DC é a utilização de um circuito conhecido como Ponte H. Neste tipo de circuito a partir dos sinais gerados do micro controlador é possível controlar a direção de um motor DC. Devido à disposição dos seus componentes, facilita o modo de selecionar o sentido da rotação de um motor, apenas invertendo a polaridade sobre seus terminais. Também é utilizado em circuitos digitais, pois como os sinais de saída dos microcontroladores não suportam a corrente necessária e nem possuem a tensão adequada para acionar um motor, é necessária uma unidade de potência que possa alimentá-lo convenientemente.

Neste projeto foi utilizado o circuito integrado modelo L293C que tem 20 pinos e 4 canais (ver Datasheet no Anexo I) . Este CI foi escolhido pois pode facilmente ser utilizado como uma ponte-H dupla, desta forma pode controlar 2 motores com um único CI.

REGULADOR DE TENSÂO

A placa arduino pode funcionar com uma fonte de alimentação externa de 6 a 20 volts. No entanto se a alimentação for inferior a 7V, o pino de 5V pode fornecer menos de cinco volts e a placa pode se mostrar instável, ou se a alimentação for maior que 12V o regulador de voltagem pode superaquecer e danificar a placa.

O microcontrolador opera apenas a 5V, para utilizar tensões superiores sem danificar este componente faz-se necessário o uso desse regulador de voltagem (imagem 6), que dependendo do modelo até parece um transistor, ele reduz a tensão de entrada para os 5V necessários ao microcontrolador. Este componente possui proteções contra curto-circuito e excesso de temperatura e sua saída será sempre 5V, independente se houver flutuação na entrada. Com esse regulador de voltagem fornecido pela placa arduino, utilizaremos uma fonte de alimentação de 9V no projeto.

Imagem 6: Detalhe do CI com função de regular a voltagem no Arduino

ESTABILIZAÇÂO DA CORRENTE DO MOTOR

Em paralelo ao motor é necessário adicionar um capacitor, para filtrar o ruido das escovas do motor e para estabilizar a corrente do motor ao reveter a corrente.

Os capacitores são conectados na porta da saída inversora, pois tem como principal função fornecer potência reativa necessária para a comutação dos tiristores da ponte inversora. Eles garantem os níveis e as formas de onda compatíveis com o funcionamento do motor.

A estabilização é feita em função do ruído, pois eliminando o ruído que o motor induz na rede, a corrente fica estável , causando assim a estabilização necessária.

RESISTOR PULL-UP

Para leitura da rota definida e do detector de colisão, é utilizado as entradas digitais em conjunto com resistores pull-up internos para evitar flutuação em pinos configurados como entradas(INPUT).

Eles são usados em projetos de circuitos lógicos eletrônicos para garantir que entradas para sistemas lógicos se ajustem em níveis lógicos esperados. Em geral, é necessário implementar externamente, mas muitas vezes há pull-ups implementados

internamente em alguns pinos do microcontrolador. No caso do Arduino, existe em sua placa resistores de aproximadamente 20KOhms que poderão ser ativados para funcionarem como pull-ups internos para todos os pinos digitais e analógicos (com uma tensão de 1,7v), portanto não há necessidade de implementar um pull-up externamente nesse projeto.

CIRCUITO CONTROLADOR

Após assimilado o conceito de ponte H, regularizado a voltagem e estabilizado a corrente do motor, foi projetado o circuito que irá controlar o carro (imagem 7 e 8). Esse circuito possui duas alimentações, uma para o circuito em conjunto com o motor dianteiro (bateria de 9v) e outra adicional exclusiva para o motor traseiro (3 pilhas AA, 4,5v), o qual consome mais energia. Para que o controle dos motores seja realizado de forma automática, o Arduino foi programado de acordo com o algoritmo implementado (ver apêndice I), nele o controlador é mantido em loop e para cada passagem é lido os estados do DIP Switch para escolha da rota pré-definidas, sendo elas:

- 00: Desligado
- 01: Tração para frente e direção para a direita, logo, anda em círculos;
- 10: O carro manobra, indo para frente e direita, e depois para trás e esquerda, por 1,5s em cada ciclo;
- 11: A rota é definida de modo aleatório, sendo gerado um número para definir a tração e direção. Quando é detectado uma colisão, o LED ficará acesso e o carro traciona no sentido contrário, com uma direção aleatória;

Imagem 7: Montagem do Circuito Controlador

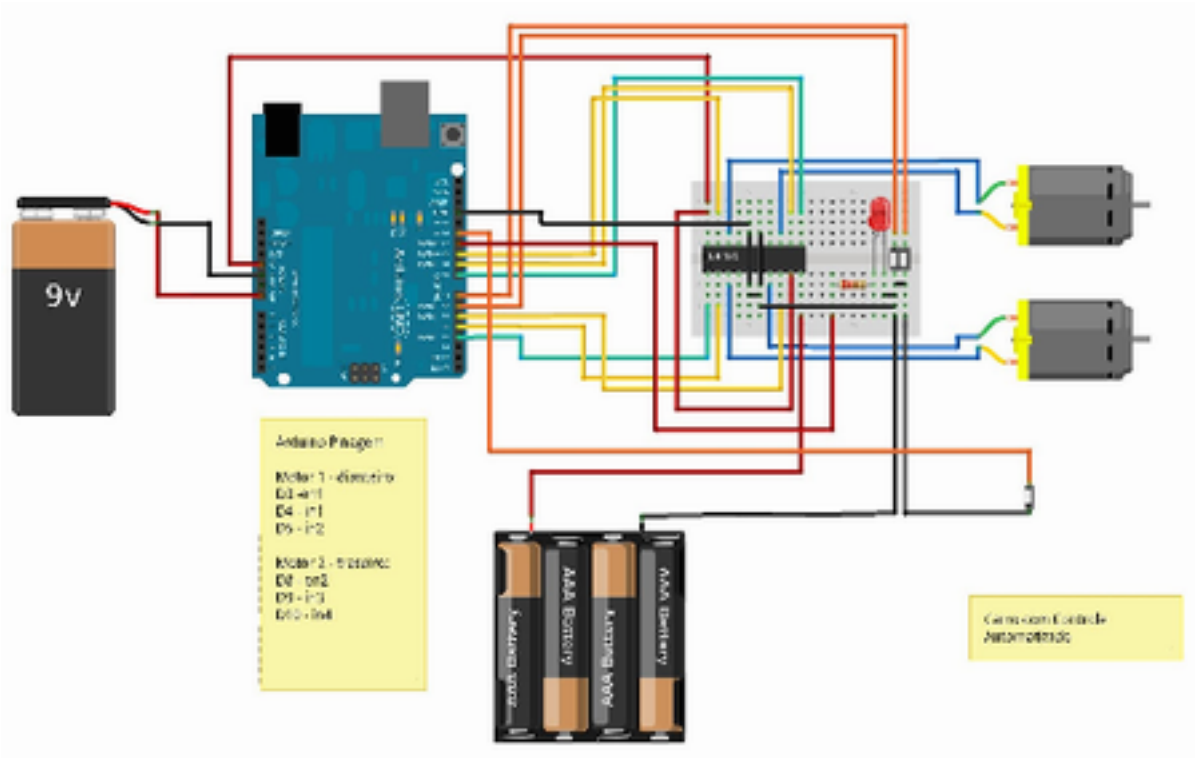
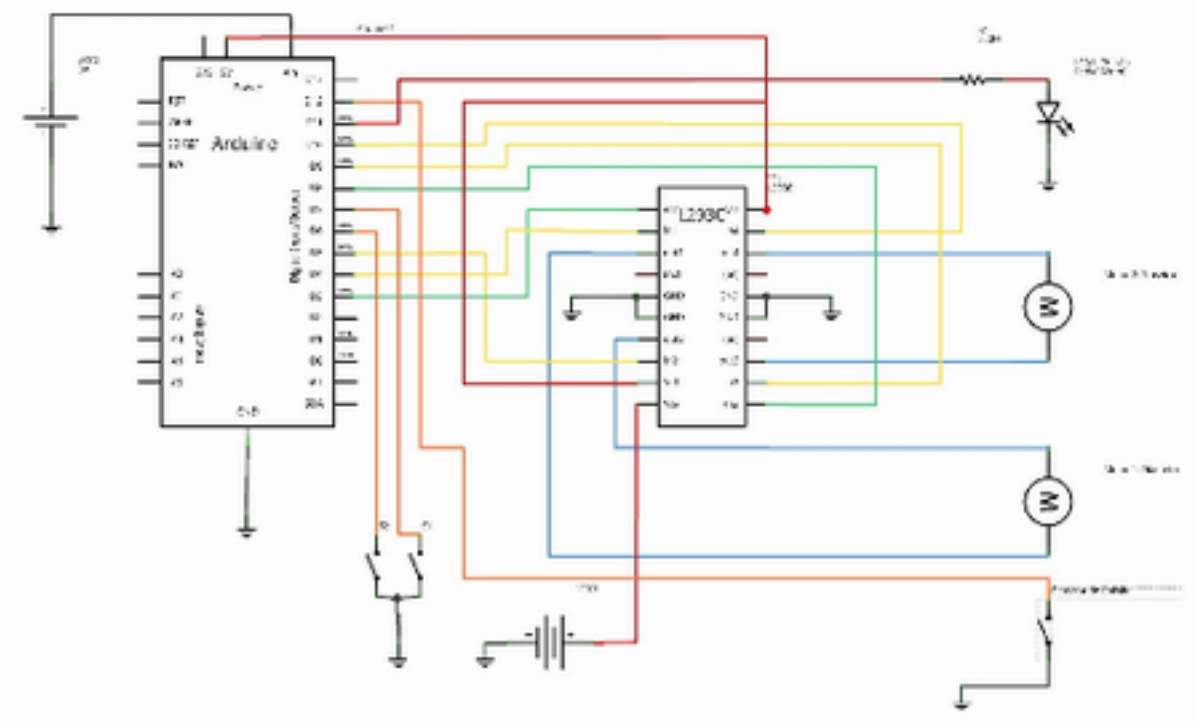


Imagem 8: Esquema do Circuito Controlador



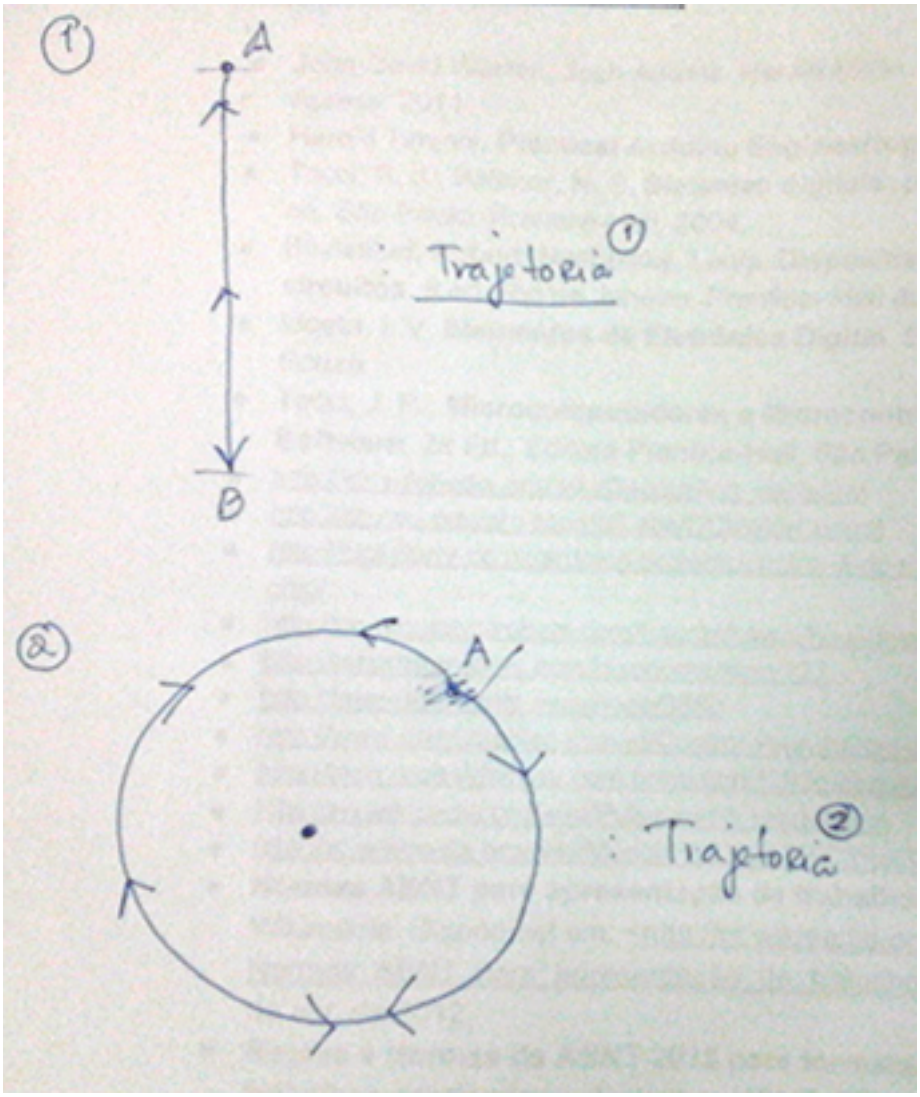
PROVA DE CONCEITO

Com a finalidade de determinar se o protótipo foi implementado de forma correta, o orientador propôs dois testes de trajetórias adicionais (*imagem 9*), sendo elas:

1. Carro vai do ponto *A* ao ponto *B* (2,5 m) e retorna ao ponto *A* em linha reta continuamente;

2. Carro vai do ponto *A* ao ponto *A*, em rota circular, e depois retorna do ponto *A* ao ponto *A* (de ré) e assim sucessivamente;

Imagem 9: Rotas para a prova de conceito definida pelo orientador do projeto.



Cálculo para a trajetória 1

Para a o calculo da trajetória 1, foi necessário calcular a velocidade que o carro desenvolve, para isso foi definido uma rota auxiliar (rota 4 do algoritmo), onde o carro anda por 1s e para, e com isso foi mensurado o espaço percorrido igual a 0,9m. tendo como solução a velocidade:

$$v = \Delta s / \Delta t = 0.9m / 1s = 0.9\,m/s$$

Considerando a aceleração como constante e instantânea, o espaço de 2,0 m será percorrido em 1,8s:

$$t = \Delta s / v = 2m / 0{,}9ms / s = 1{,}8\,s$$

Logo, bastou programar uma nova rota 5 com o tempo de 1,8s, para que a trajetória 1 seja realizada.

Cálculo para a trajetória 2

Para o calculo da trajetória 2 observou-se que quando o carro estava em rota circular, segundo a rota 2 previamente definida, o carro desenvolvia uma circunferência em torno de uma grade formada por um grid de 3x3 pisos (*imagem 10*), sendo assim bastou calcular a diagonal desse quadrante, e tomando ele como diâmentro da circunferência, chegou-se a conclusão de que o perímetro do mesmo é de 3,99m:

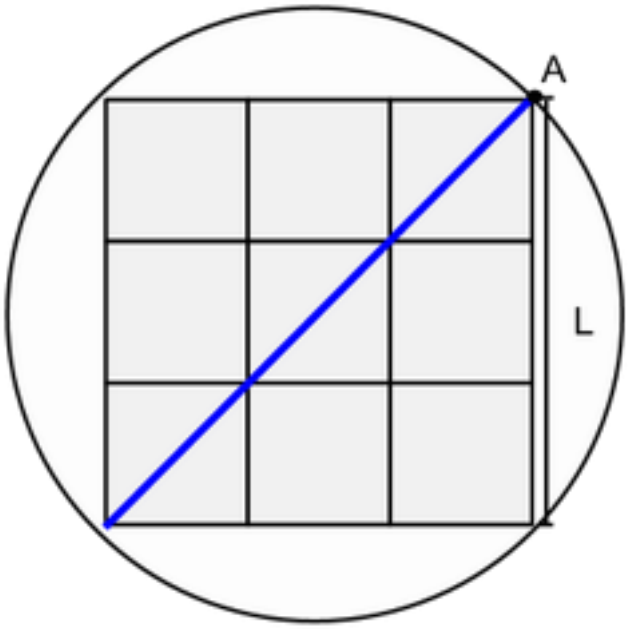
Considerando:

- L: $3 . 0{,}30m = 0{,}9m$
- D: diâmetro da circunferência;
- P: perímetro da circunferência;

Portanto:

$$D = L.\sqrt{2} = 1{,}27m$$
$$P = \pi . D = 3{,}99m$$
$$t = \Delta s / v = 3{,}99m \ / \ 0{,}9m/s = 4{,}44s$$

Imagem 10: Cálculo do perímetro.



Logo, foi programado o algoritmo 6 com o tempo de 4,44s com a finalidade de percorrer a trajetória 2.

APÊNDICE I: ALGORITMO DE CONTROLE

```
//PROJETO CARRO MICROCONTROLADO: Com Rotas Pré-Definidas e Aleatórias
//Projeto da disciplina: Projeto Integrado II
//Outubro de 2012

//define portas de controle:
//direcao
#define dir_en 2
#define dir_es 3
#define dir_di 5
//tracao
#define tra_en 8
#define tra_fr 9
#define tra_tr 10
//led
#define led 11
//DIP Switch
#define dip_s1 7
#define dip_s2 6
//Detector de Colisao
#define cd_wire 12

//define o tempo de parada para as manobras
long t_m = 250;
//define o tempo de tracao
long t_t = 1500;

int estado = 0;
int s1,s2,cd = 0;
long random_dir, random_time;

void setup() {
  Serial.begin(9600);

  //define IO
  //motor dianteiro
  pinMode(dir_en, OUTPUT); //enable motor 1
  pinMode(dir_es, OUTPUT);
  pinMode(dir_di, OUTPUT);

  //motor traseiro
  pinMode(tra_en, OUTPUT); //enable motor 2
  pinMode(tra_fr, OUTPUT);
  pinMode(tra_tr, OUTPUT);

  //led colisao
  pinMode(led, OUTPUT);

  //detector colisao
  pinMode(cd_wire, INPUT);

  //DIP Switch
  pinMode(dip_s2, INPUT);
  pinMode(dip_s1, INPUT);

  //Ativa os resistores internos no modo pullup
  digitalWrite(dip_s1, HIGH);
  digitalWrite(dip_s2, HIGH);
  digitalWrite(cd_wire, HIGH);

  //inicializa o random utilizando um numero do ruido da entrada no pino 0
  randomSeed(analogRead(0));

}

void loop() {
```

```
  turn_off();
  delay(10);
  s1 = digitalRead(dip_s1);
  s2 = digitalRead(dip_s2);
  cd = digitalRead(cd_wire);
  //imprime as leituras dos pullups
  Serial.print("s1: ");
  Serial.print(s1);
  Serial.print("\t");
  Serial.print("s2: ");
  Serial.print(s2);
  Serial.print("\t");
  Serial.print("cd: ");
  Serial.println(cd);
  Serial.println("---------------------");

  if(s1==LOW && s2==HIGH && cd==HIGH){
    rota_1();
  }
  else if(s2==LOW && s1==HIGH && cd==HIGH){
    rota_2();
  }
  else if(s2==LOW && s1==LOW){
    rota_3();
  }
  else  if(cd==LOW){
    //colisao detectada, led aceso
    digitalWrite(led, HIGH);
    delay(t_t / 2);
  }
  else{
    //aguardando rota, led pisca por 50ms
    digitalWrite(led, HIGH);
    delay(50);
    digitalWrite(led, LOW);
    delay(950);
  }
}

//manobra o carro
void rota_1(){
  //necessario dobro do tempo de manobra, para nao sobrecarregar o motor
  //uma vez uqe o carro vai para frente e para tras
  delay(t_m * 2);
  digitalWrite(tra_en, HIGH);
  digitalWrite(dir_en, HIGH);
  if(estado==0){
    estado = 1;
                digitalWrite(dir_di, HIGH);
                delay(t_m);
    digitalWrite(tra_fr, HIGH); //frente
  }
  else{
    estado = 0;
    digitalWrite(dir_es, HIGH);
                delay(t_m);
    digitalWrite(tra_tr, HIGH); //ré
  }
  delay(t_t);
}

//roda para a direita
void rota_2(){
  digitalWrite(tra_en, HIGH);
  digitalWrite(dir_en, HIGH);

  digitalWrite(tra_fr, HIGH); //frente
  digitalWrite(dir_di, HIGH);

  delay(t_t);
}
```

```
//desliga todas as saidas
void turn_off(){
  digitalWrite(tra_fr, LOW);
  digitalWrite(tra_tr, LOW);
  digitalWrite(dir_es, LOW);
  digitalWrite(dir_di, LOW);
  digitalWrite(led, LOW);
  digitalWrite(tra_en, LOW);
  digitalWrite(dir_en, LOW);
}

//aleatorio para busca
void rota_3(){
  delay(t_m);
  digitalWrite(tra_en, HIGH);
  digitalWrite(dir_en, HIGH);
  //gera numeros randomicos para a rota
  // 7 8 9
  // 4 5 6
  // 1 2 3
  if(cd==HIGH){
    random_dir = random(7, 10);
    random_time = random((t_t) / 2, t_t);
  }
  else if(cd==LOW){
    //colisao detectada, led aceso, anda para tras por 2s
    digitalWrite(led, HIGH);
    random_dir = random(1, 4);
                //volta 120% do tempo max de tracao
    random_time = (t_t * 1.2);
    delay(t_m);
  }
  switch(random_dir){
  case 1:
    digitalWrite(dir_es, HIGH);
    delay(t_m);
    digitalWrite(tra_tr, HIGH);
    break;
  case 2:
    digitalWrite(tra_tr, HIGH);
    break;
  case 3:
    digitalWrite(dir_di, HIGH);
    delay(t_m);
    digitalWrite(tra_tr, HIGH);
    break;

  case 7:
    digitalWrite(dir_es, HIGH);
    delay(t_m);
    digitalWrite(tra_fr, HIGH);
    break;
  case 8:
    digitalWrite(tra_fr, HIGH);
    break;
  case 9:
    digitalWrite(dir_di, HIGH);
    delay(t_m);
    digitalWrite(tra_fr, HIGH);
    break;

  default:
    digitalWrite(led, HIGH);
    delay(t_m);
    digitalWrite(led, LOW);
    delay(t_m);
    break;
  }
  delay(random_time);
}
```

```
//rota de apoio para calculo da velocidade
void rota_aux_4(){
  delay(t_m * 2);
  digitalWrite(tra_en, HIGH);
  digitalWrite(tra_fr, HIGH); //frente
  delay(1000);
  digitalWrite(tra_en, LOW); //desliga
  delay(3000); //aguarda 3s para medir
}

//prova de conceito - trajetoria 1
void rota_5(){
  delay(t_m * 2);
  digitalWrite(tra_en, HIGH);
  if(estado==0){
    estado = 1;
    digitalWrite(tra_fr, HIGH); //frente
  }else{
    estado = 0;
    digitalWrite(tra_tr, HIGH); //ré
  }
  delay(2500); //tempo para percorrer 3m
}

//prova de conceito - trajetoria 2
void rota_6(){
  delay(t_m * 2);
  digitalWrite(tra_en, HIGH);
  digitalWrite(dir_en, HIGH);
  digitalWrite(dir_di, HIGH); //roda sempre para direita
  delay(t_m);
  if(estado==0){
    estado = 1;
    digitalWrite(tra_fr, HIGH); //frente
  }else{
    estado = 0;
    digitalWrite(tra_tr, HIGH); //ré
  }
  delay(4580); //tempo para percorrer o perimetro com 5,49m
}
```

ANEXO I: L293C DATASHEET

SGS-THOMSON MICROELECTRONICS

L293C

PUSH-PULL FOUR CHANNEL/DUAL H-BRIDGE DRIVER

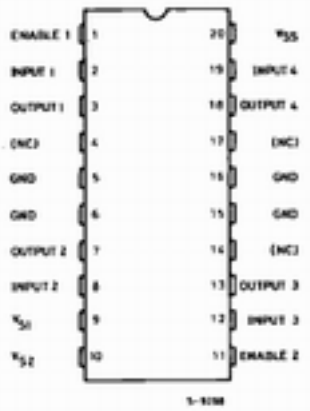
PRELIMINARY DATA

- 600 mA OUTPUT CURRENT CAPABILITY PER CHANNEL
- 1.2 A PEAK OUTPUT CURRENT (non repetitive) PER CHANNEL
- ENABLE FACILITY
- OVER TEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (high noise immunity)
- SEPARATE HIGH VOLTAGE POWER SUPPLY (up to 44 V)

POWERDIP (16 + 2 + 2)

ORDERING NUMBER : L293C

PIN CONNECTION



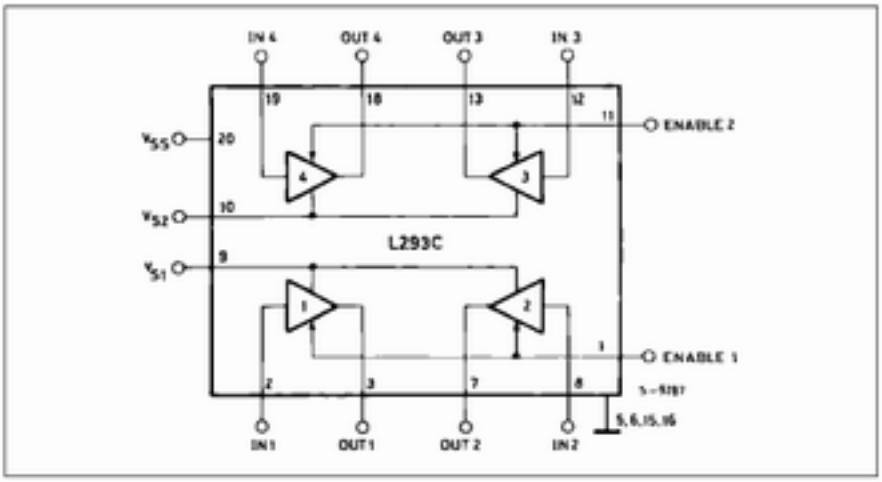
DESCRIPTION

The L293C is a monolithic high voltage, high current integrated circuit four channel driver in a 20 pin DIP. It is designed to accept standard TTL or DTL input logic levels and drive inductive loads (such as relays, solenoids, DC and stepping motors) and switching power transistors.

The device may easily be used as a dual H-bridge driver: separate chip enable and high voltage power supply pins are provided for each H-bridge. In addition, a separate power supply is provided for the logic section of the device.

The L293C is assembled in a 20 lead plastic package which has 4 center pins connected together and used for heatsinking.

BLOCK DIAGRAM

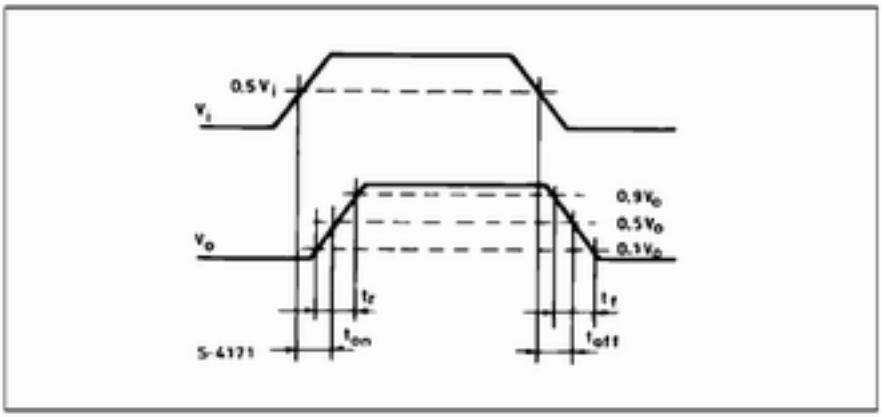


TRUTH TABLE

Input	Enable	Output
H	H	H
L	H	L
X	L	Z

Z = High output impedance

SWITCHING TIMES



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Supply Voltage	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i	Input Voltage	7	V
V_{EN}	Enable Voltage	7	V
I_O	Peak Output Current (non-repetitive t = 5 ms)	1.2	A
P_{tot}	Total Power Dissipation at $T_{pin\ 4,\ 5,\ 12,\ 13\ case}$ = 80°C	5	W
T_{stg}, T_j	Storage and Junction Temperature	−40 to 150	°C

THERMAL DATA

Symbol	Parameter		Value	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	14	°C/Ω
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	80	°C/Ω

ELECTRICAL CHARACTERISTICS

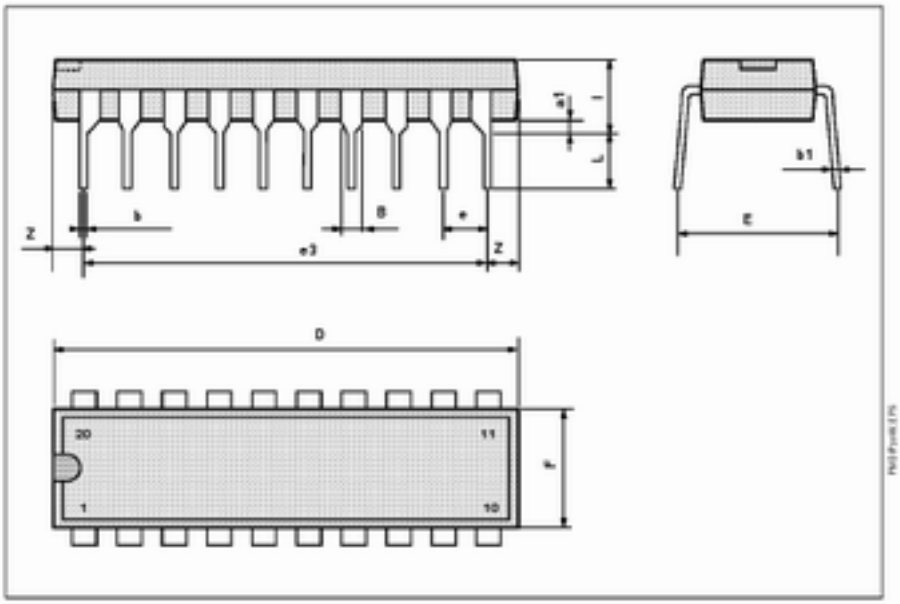
(for each channel, V_S = 24 V, V_{SS} = 5 V, T_{amb} = 25°C, unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 9, 10)		V_{SS}		44	V
V_{SS}	Logic Supply Voltage (pin 20)		4.5		7	V
I_S	Total Quiescent Supply Current (pin 9, 10)	V_i = L I_O = 0 V_{EN} = H V_i = H I_O = 0 V_{EN} = H V_{EN} = L		2 16	6 24 4	mA
I_{SS}	Total Quiescent Logic Supply Current (pin 20)	V_i = L I_O = 0 V_{EN} = H V_i = H I_O = 0 V_{EN} = H V_{EN} = L		44 16 16	60 22 24	mA
V_{iL}	Input Low Voltage (pin 2, 8, 12, 19)		-0.3		1.5	V
V_{iH}	Input High Voltage (pin 2, 8, 12, 19)		2.3		V_{SS}	V
I_{iL}	Low Voltage Input Current (pin 2, 8, 12, 19)	V_i = 1.5 V			-10	µA
I_{iH}	High Voltage Input Current (pin 2, 8, 12, 19)	2.3 V ≤ V_i ≤ V_{SS} − 0.6 V		30	100	µA
$V_{EN\ L}$	Enable Low Voltage (pin 1, 11)		-0.3		1.5	V
$V_{EN\ H}$	Enable High Voltage (pin 1, 11)		2.3		V_{SS}	V
$I_{EN\ L}$	Low Voltage Enable Current (pin 1, 11)	$V_{EN\ L}$ = 1.5 V		-30	-100	µA
$I_{EN\ H}$	High Voltage Enable Current (pin 1, 11)	2.3 V ≤ $V_{EN\ H}$ ≤ V_{SS} − 0.6			±10	µA
$V_{CE\ (sat)H}$	Source Output Saturation Voltage (pins 3, 7, 13, 18)	I_O = − 0.6 A		1.4	1.8	V
$V_{CE\ (sat)L}$	Sink Output Saturation Voltage (pins 3, 7, 13, 18)	I_O = + 0.6 A		1.2	1.8	V
t_r	Rise Time (*)	0.1 to 0.9 V_O		250		ns
t_f	Fall Time (*)	0.9 to 0.1 V_O		250		ns
t_{on}	Turn-on Delay (*)	0.5 V_i to 0.5 V_O		750		ns
t_{off}	Turn-off Delay (*)	0.5 V_i to 0.5 V_O		200		ns

(*) See switching times diagram.

POWERDIP (16 + 2 +2) PACKAGE MECHANICAL DATA

Dimensions	Millimeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
a1	0.51			0.020		
B	0.85		1.4	0.033		0.055
b		0.5			0.020	
b1	0.38		0.5	0.015		0.020
D			24.8			0.976
E		8.8			0.346	
e		2.54			0.100	
e3		22.86			0.900	
F			7.1			0.280
i			5.1			0.201
L		3.3			0.130	
Z			1.27			0.050



REFERÊNCIAS

- John-David Warren, Josh Adams, Harald Molle. **Arduino Robotics**. Ed. Apress, 2011.
- Harold Timmis. **Practical Arduino Engineering**. Ed. Apress, 2011.
- Tocci, R. J.; Widmer, **N. S. Sistemas digitais: princípios e aplicações**. 8. ed. São Paulo: Prentice-Hall, 2004.
- Boylestad, Robert; Nashelsky, Louis. **Dispositivos eletrônicos e teoria de circuitos**. 8 ed. Rio de Janeiro: Prentice- Hall do Brasil, 2004. 649 p.
- Idoeta, I. V. **Elementos de Eletrônica Digital**. São Paulo: Érica, 2006. 38 ed Souza.
- Tocci, J. R., **Microcomputadores e Microcontroladores - Hardware e Software**, 2a Ed., Editora Prentice-Hall, São Paulo:1983.
- http://en.wikipedia.org/wiki/Decoupling_capacitor
- http://itp.nyu.edu/physcomp/Labs/DCMotorControl
- http://luckylarry.co.uk/arduino-projects/control-a-dc-motor-with-arduino-and-l293d-chip/
- http://communityofrobots.com/tutorial/kawal/how-drive-dc-motor-using-l293d-arduino
- http://letsmakerobots.com/taxonomy/term/127
- http://letsmakerobots.com/node/3880
- http://www.instructables.com/id/Control-your-motors-with-L293D-and-Arduino/
- http://blog.repeatdomiau.com.br/miadas/l293d-circuito-integrado-de-dupla-ponte-h
- http://en.wikipedia.org/wiki/Pulse-width_modulation
- http://pt.wikipedia.org/wiki/Modula%C3%A7%C3%A3o_por_largura_de_pulso
- **Normas ABNT para apresentação de trabalhos científicos.** Wikimedia. Disponível em: <http://pt.wikipedia.org/wiki/Normas_ABNT_para_apresentação_de_trabalhos_científicos>. Acesso em 04 de abr. de 2012.
- **Regras e Normas da ABNT 2012 para formatação de trabalhos acadêmicos.** Trabalhos ABNT. Disponível em: <http://www.trabalhosabnt.com/regras-normas-abnt-formatacao>. Acesso em 04 de abr. de 2012.
- **Trabalhos acadêmicos: Normas da ABNT.** Firb. Disponível em: <http://www.firb.br/abntmonograf.htm>. Acesso em 04 de abr. de 2012.