UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO

MARCOS VINÍCIUS ALVES COSTA

CONSTRUÇÃO DE UMA PLATAFORMA DE BAIXO CUSTO PARA APLICAÇÃO EM ROBÓTICA EDUCACIONAL

MOSSORÓ - RN
2013

MARCOS VINÍCIUS ALVES COSTA

CONSTRUÇÃO DE UMA PLATAFORMA DE BAIXO CUSTO PARA APLICAÇÃO EM ROBÓTICA EDUCACIONAL

Monografia apresentada à Universidade Federal Rural do Semi-Árido (UFERSA), como exigência final para obtenção do título de Engenheiro de Energia.

Orientador: Prof. Dr. Francisco José Targino Vidal - UFERSA.

MOSSORÓ - RN
2013

Ficha catalográfica preparada pelo setor de classificação e catalogação da Biblioteca "Orlando Teixeira" da UFERSA

C837c Costa, Marcos Vinicius Alves.

Construção de uma plataforma de baixo custo para aplicação em robótica educacional / Marcos Vinícius Alves Costa. – Mossoró, RN : 2013.

70f. : il.

Orientador: Profº. Dr. Francisco José Targino Vidal.

Monografia (Graduação) – Universidade Federal Rural do Semi-Árido, Graduação em Engenharia de Energia, 2013.

1. Robótica e educacional. 2. Robô de baixo custo.

3. Microcontroladores. I. Título.

CDD: 629.892

Bibliotecária: Marilene Santos de Araújo

CRB-5/1033

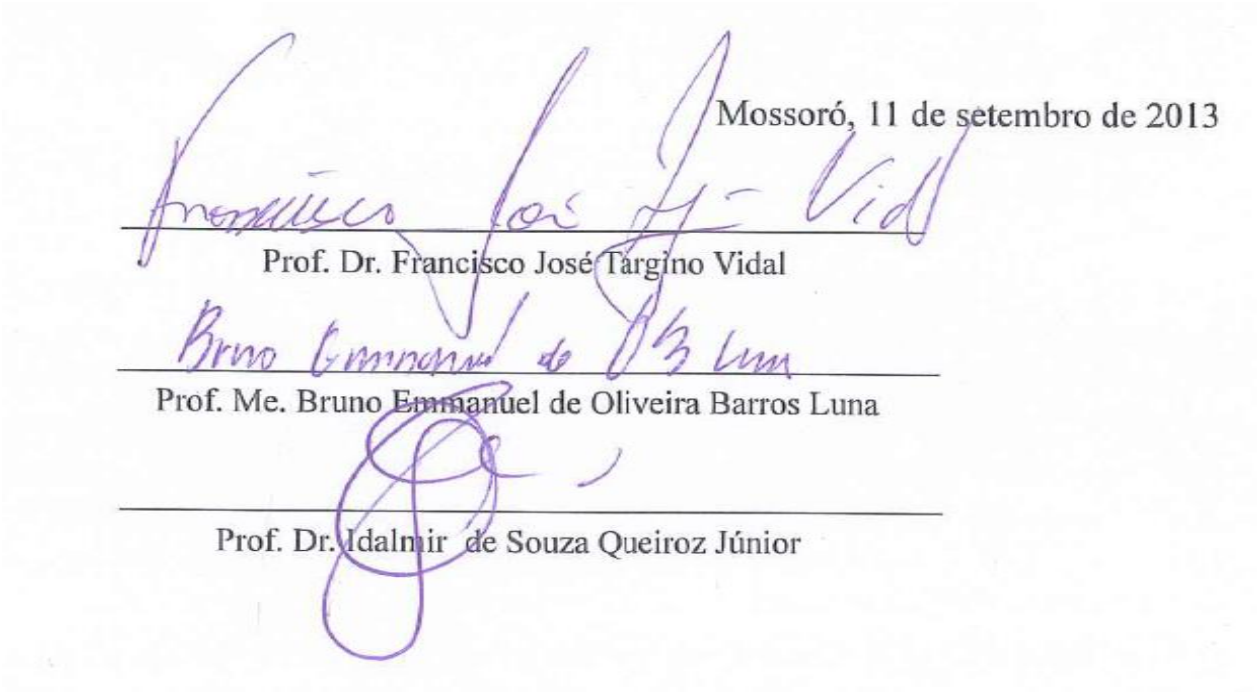
MARCOS VINÍCIUS ALVES COSTA

CONSTRUÇÃO DE UMA PLATAFORMA DE BAIXO CUSTO PARA APLICAÇÃO EM ROBÓTICA EDUCACIONAL

Monografia apresentada à Universidade Federal Rural do Semi-Árido (UFERSA), como exigência final para obtenção do título de Engenheiro de Energia.

APROVADO EM ____/____/____

BANCA EXAMINADORA



AGRADECIMENTOS

A toda minha família, na pessoa dos meus pais Francisco Firmino da Costa Filho e Maria Auxiliadora Alves Costa e minha irmã Maria Gabriela Alves Costa, que me ensinaram a viver, e que com muita confiança, dedicação, força e amor, sempre me incentivaram e cobraram em minhas decisões.

Ao meu professor orientador Francisco José Targino Vidal que me ajudou dentre outras coisas no amadurecimento dentro da profissão de engenheiro.

A todos os meus colegas de curso e amigos, principalmente a Vinícius de Paiva Costa que muito me ajudou na elaboração deste trabalho e a Antônio Elias Cornejo Pacherrez e Jefferson Romulo Rebouças pela amizade e discussões "filosóficas" de final de semana.

A Deus por sempre me guiar pelos caminhos da vida, sabendo das minhas preocupações e dando força para enfrentar os desafios.

À meus avós paternos e maternos, pela grande dedicação a família e pelo exemplo de pessoas simples, trabalhadoras e verdadeiras.

DEDICO

“Nas águas do rio da vida chega mais longe quem: nada como deve, quando deve e até onde deve”.

(Jigoro Kano)

RESUMO

A utilização da robótica como ferramenta educacional está a ganhar espaço nos diferentes níveis da educação como elemento dinamizador dos métodos de ensino, abordando o aprendizado de maneira interdisciplinar e lúdica. Dentro de sua abrangência, a robótica é constituída por ferramentas como componentes eletrônicos e eletromecânicos e programas de controle, e todos estes são hoje fatores que determinam a acessibilidade da população à utilização desta ferramenta, isto, pois os kits robóticos mais comuns comercializados no país estão em uma faixa de preço que inviabiliza a sua obtenção por parte da grande maioria. Neste cenário, foi realizada a construção de um robô de baixo custo para aplicação educacional, que consiste no controle de dois motores CC, a partir do microcontrolador PIC 16F628A. O envio de comandos é feito a partir de um computador, que se comunica via bluetooth com o robô, em um software que permite o envio de comandos por controle remoto ou por um programa de sequência de ações.

Palavras-chave: Robótica Educacional, Robô de Baixo Custo, Microcontroladores.

ABSTRACT

The use of robotics as an educational tool is gaining ground in the different levels of education as a dynamic element in teaching methods, addressing learning in an interdisciplinary and playful manner. Within its scope, robotics consists of tools such as electronic and electromechanical components and control programs, and all these are now factors that determine the accessibility of the population to use this tool, i.e., as the most common robotic kits marketed in the country are in a price range that prevents obtaining it by the majority. In this scenario, we performed the construction of a low-cost robot for educational application, which consists in controlling two DC motors from the PIC microcontroller 16F628A. Sending commands is done from a computer that communicates with the robot via bluetooth in a software that allows you to send commands by remote control or by a program of sequence of actions.

Key words: Educational Robotics, Low-Cost Robot, Microcontrollers.

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A robótica e suas aplicações encontram-se em plena expansão no cenário mundial sendo uma das práticas mais representativas de novas tecnologias. É possível observar a sua utilização nos mais diversos âmbitos sociais, seja na indústria, empregado com a finalidade de melhorar a produtividade, seja em substituição do trabalho humano em áreas de risco a saúde, até aplicações educacionais como de interesse deste trabalho.

O uso da robótica como instrumento no processo de ensino-aprendizagem provou ser uma forte aliada no processo de aquisição do conhecimento, pois possibilita estimular o pré-projeto, a engenharia e habilidades em computação, caracterizando a atividade robótica como interdisciplinar sendo, por isso, altamente relevante para o currículo escolar (Silvana, 2004).

Os robôs a serem utilizados educacionalmente, são da chamada segunda geração de robôs. Estes são dotados de sensores externos e internos (sensores de luz, de toque, peso, etc.), a programação adotada permite que se adequem as situações nas quais tais dispositivos se encontram, possibilitando a interação do robô com o meio onde se desloca ou se movimenta.

Para a prática da robótica educacional são geralmente utilizados os kits didáticos para robótica. Alguns desses kits são distribuídos comercialmente, podendo em alguns casos possuir várias peças para encaixe ou mesmo permitindo apenas a alteração na sua programação e lógica de controle. Em contra partida estes kits chegam ao mercado com preços pouco acessíveis, no Brasil principalmente por causa dos impostos de importação sobre o produto. Desta forma, a utilização desses kits em larga escala dentro das escolas públicas torna-se muito difícil.

2 REVISÃO DE LITERATURA

Como dito em [AZEVEDO, AGLAÉ, PITTA]: Certamente muitas pessoas conseguem identificar um dispositivo robótico, mas terão alguma dificuldade em construir uma definição abrangente sobre o mesmo. Pensando desta forma, Joseph F. Engelberger, considerado o pai da robótica por construir e vender o primeiro robô industrial mencionou, em certa ocasião, seu entendimento acerca do que seria um robô em uma única frase: “Eu não posso definir um robô, mas eu reconheço um quando o vejo”.

Segundo o dicionário Aurélio (versão on-line), robô é um “aparelho automático, geralmente em forma de boneco, que é capaz de cumprir determinadas tarefas. / Fig. Pessoa que procede como um robô, isto é, que executa ordens sem pensar”.

Podemos também utilizar a definição do R.I.A. (*Robotics Industries Association*), o qual nos coloca que: “Robô é um manipulador re-programável e multifuncional projetado para mover materiais, partes, ferramentas ou dispositivos especializados através de movimentos variáveis programados para desempenhar uma variedade de tarefas”.

Observando as definições anteriores, nota-se que robôs são projetados para auxiliar o homem nas mais diversas tarefas cotidianas, sendo utilizada dentro das indústrias ou até mesmo em atividades de risco a saúde humana, bem como para fins educacionais.

A robótica é uma área responsável pelo desenvolvimento de dispositivos capazes de realizar tarefas com eficiência e exatidão, incluindo as que são impossíveis de serem executadas pelo homem sem risco de vida. Ela busca o desenvolvimento e a integração de técnicas e algoritmos para a criação de robôs (PAZOS, 2002).

A origem da robótica educacional se dá no início da década de 50, quando o neurologista William Grey Walter construiu robôs móveis que denominou de tartarugas, com os nomes de Elmer e Elsie. Cada tartaruga tinha um conjunto de dois comportamentos elementares: no primeiro a tartaruga após colidir com um obstáculo afastava-se dele, e no segundo, ela se aproximava de fontes de luz, exceto se a fonte de luz fosse muito forte, caso em que se afastava. Embora limitados a estes comportamentos os pequenos robôs eram capazes, ainda assim, de exibir uma interação

complexa com o ambiente e entre si, por exemplo, quando um deles possuía uma fonte de luz (GONÇALVES, 2007).

Seymour Papert foi pioneiro em utilizar a robótica para fins educacionais. Ele desenvolveu a linguagem Logo no MIT, tendo, a princípio, criado uma tartaruga de solo para utilização da versão inicial da linguagem Logo, inspirado nas tartarugas desenvolvidas por Grey Walter. Era um dispositivo móvel pequeno que poderia ser controlado através de comandos do computador. Ele pretendia que a tartaruga fosse um objeto no qual as crianças poderiam usar o conhecimento de seu próprio corpo para compreender o movimento da tartaruga, podendo, deste modo, a tartaruga se tornar um "objeto para pensar com" (thing to think with). Com o surgimento de computadores pessoais, a tartaruga de solo deu lugar a uma tartaruga virtual, que se movimentava na tela do computador (MARTIN, 1988).

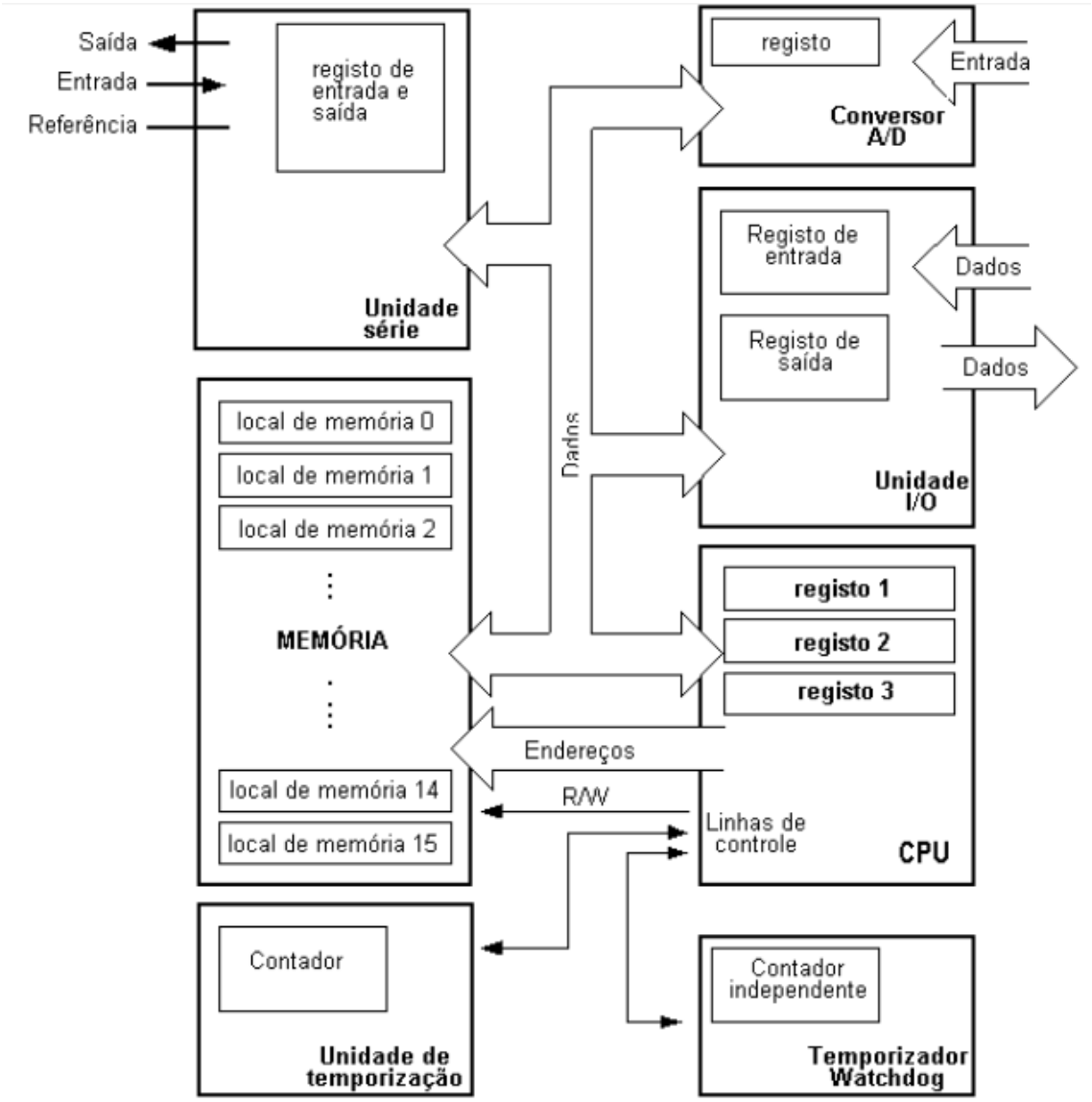
Como apresentado em [PAZOS, 2002], a robótica educacional é uma ferramenta de grande potencial para o processo de ensino aprendizagem. É uma proposta educativa que vêm de encontro com as teorias e visões dos mais conceituados educadores da atualidade. Com a robótica educacional, além de se ter o desenvolvimento da inteligência lógico matemática que é mais evidente, pelo fato de trabalhar com linguagem de programação e cálculos necessários para a construção dos modelos, promove o desenvolvimento da inteligência linguística, intrapessoal e até da espacial, pois envolve aspectos como trabalhar em grupo, planejamento de ações, construção de modelos e apresentação do resultado final.

2.1. MICROCONTROLADORES

Os microcontroladores têm os seus primórdios no desenvolvimento da tecnologia dos circuitos integrados. Este desenvolvimento tornou possível armazenar centenas de milhares de transístores num único chip. Isso constituiu um pré-requisito para a produção de microprocessadores e, os primeiros computadores foram construídos adicionando periféricos externos tais como memória, portas de entrada e saída, temporizadores e outros. Um crescente aumento no nível de integração permitiu o aparecimento de circuitos integrados contendo simultaneamente processador e periféricos.

Os microcontroladores são computadores digitais integrados em um chip que tem um microprocessador ou unidade de processamento central (CPU), uma memoria para armazenar o programa, uma memória para armazenar dados e portas de entrada saída. O funcionamento dos microcontroladores é determinado pelo programa armazenado em sua memória, podendo ser escrito em diferentes linguagens de programação, além disso, os microcontroladores podem ser reprogramados repetidas vezes. Na figura 1 segue um esquemático dos elementos básicos de um microcontrolador.

Figura 1 – Diagrama de um microcontrolador com os seus elementos básicos e ligações internas.

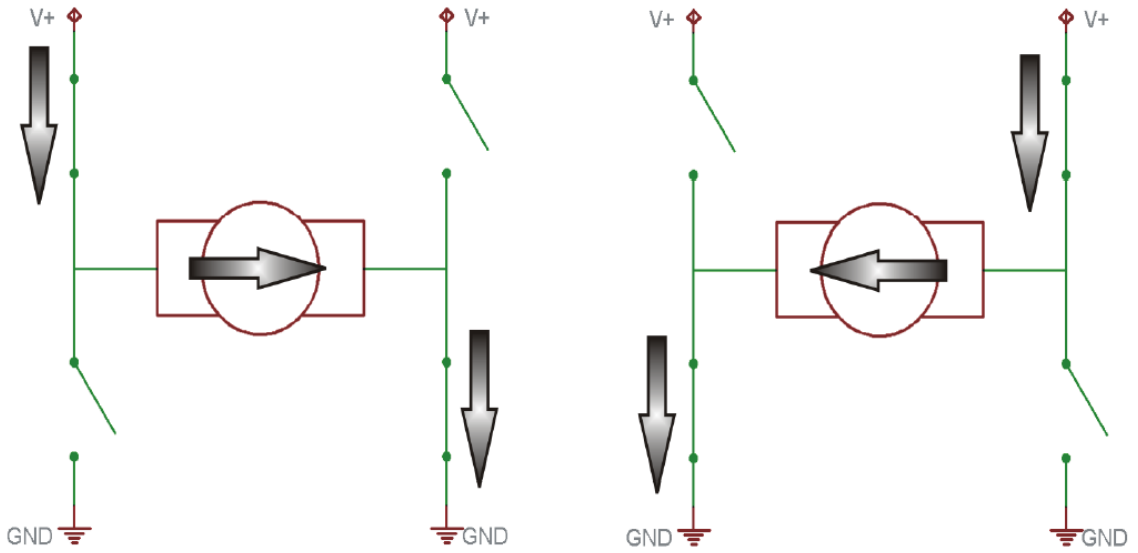


2.2. PONTE H

A ponte H é um circuito utilizado para controlar um motor DC a partir de sinais gerados por um microcontrolador. Devido à disposição dos seus componentes, torna-se extremamente fácil selecionar o sentido da rotação de um motor, apenas invertendo a polaridade sobre seus terminais. Também é importante para a utilização com 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.

Quando ligamos um motor DC com uma bateria, observamos que ele gira numa velocidade constante e numa única direção. Para alterarmos o sentido da rotação do motor, basta apenas ligar os terminais do motor de forma invertida. Para que não seja necessário fazer essa operação manualmente, podemos utilizar uma ponte H. Pode-se criá-la facilmente com a finalidade de controlar o sentido da rotação de um motor utilizando chaves simples, relés ou transistores, bastando apenas entender o seu funcionamento. Uma ponte H básica é composta por quatro chaves mecânicas ou eletrônicas posicionadas formando a letra “H”, sendo que cada uma localiza-se num extremo e o motor é posicionado no meio.

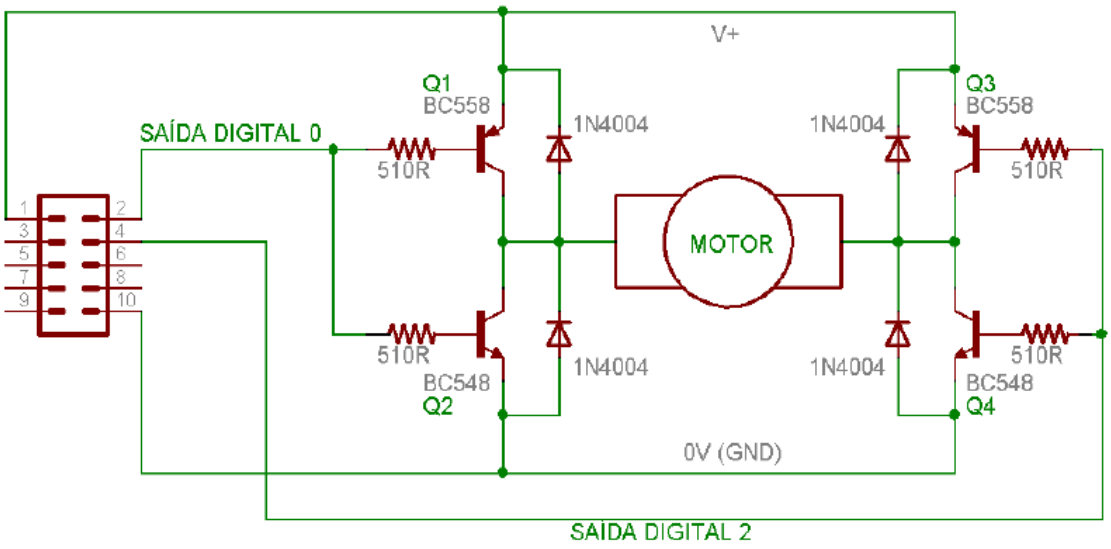
Figura 2 – Representação do uso de uma ponte H



Dentre as chaves eletrônicas, o uso de transistores é o mais conveniente no acionamento do sentido de rotação, devido a sua funcionalidade e fácil aplicação. Quando a Base do transistor é devidamente polarizada, ele é capaz de conduzir uma corrente entre seus terminais Coletor e Emissor. No caso dos transistores NPN, a condução da corrente se dará do Coletor para o Emissor, enquanto que nos transistores PNP, a corrente será conduzida do Emissor para o Coletor.

O uso de transistores também é recomendável devido às características dos sinais das saídas digitais, cujo nível de tensão é de 3,3V e a corrente máxima e de aproximadamente 20mA.

Figura 3 – Esquemático completo de uma ponte H controlando um motor DC



É importante dizer que quando os transistores são desligados, interrompendo a passagem de corrente do circuito, as propriedades indutivas do motor forçam a corrente a continuar fluindo, o que pode danificar os transistores. Para evitar possíveis danos, é adicionado um diodo em paralelo com cada transistor, com a finalidade de drenar a corrente que poderia forçar a passagem através dos transistores.

2.3. COMUNICAÇÃO BLUETOOTH

Bluetooth é um padrão global de comunicação sem fio e de baixo consumo de energia que permite a transmissão de dados entre dispositivos, desde que um esteja próximo do outro. Uma combinação de hardware e software é utilizada para permitir que este procedimento ocorra entre os mais variados tipos de aparelhos. A transmissão de dados é feita por meio de radiofrequência, permitindo que um dispositivo detecte o outro independente de suas posições, sendo necessário apenas que ambos estejam dentro do limite de proximidade.

O Bluetooth é uma tecnologia criada para funcionar no mundo todo, razão pela qual se fez necessária a adoção de uma frequência de rádio aberta e aceita em praticamente qualquer lugar do planeta. A faixa ISM (Industrial, Scientific, Medical), que opera à frequência de 2,45 GHz, é a que mais se aproxima desta necessidade, sendo utilizada em vários países, com variações que vão de 2,4 GHz a 2,5 GHz.

Como a faixa ISM é aberta, isto é, pode ser utilizada por qualquer sistema de comunicação, é necessário garantir que o sinal do Bluetooth não sofra interferência, assimo como não a gere. O esquema de comunicação FH-CDMA (Frequency Hopping - Code-Division Multiple Access), utilizado pelo Bluetooth, permite tal proteção, já que faz com que a frequência seja dividida em vários canais. O dispositivo que estabelece a conexão muda de um canal para outro de maneira bastante rápida. Este procedimento é chamado "salto de frequência" (frequency hopping) e permite que a largura de banda da frequência seja muito pequena, diminuindo sensivelmente as chances de interferência. No Bluetooth, pode-se utilizar até 79 frequências (ou 23, dependendo do país) dentro da faixa ISM, cada uma "espaçada" da outra por intervalos de 1 MHz.

Como um dispositivo se comunicando via Bluetooth pode tanto receber quanto transmitir dados (modo full-duplex), a transmissão é alternada entre slots para transmitir e slots para receber, um esquema denominado FH/TDD (Frequency Hopping / Time Division Duplex). Estes slots são canais divididos em períodos de 625 µs (microssegundos). Cada salto de frequência deve ser ocupado por um slot, fazendo com que se tenha, em 1 segundo, 1.600 saltos.

3 OBJETIVOS

O objetivo deste trabalho é a construção de uma plataforma robótica de baixo custo, que interaja com o meio em que estiver inserido por intermédio de sensores e atuadores, oferecendo a possibilidade de ser programado de acordo com as necessidades do seu usuário programador, podendo ser utilizado para fins educacionais de maneira mais acessível.

4 MATERIAIS E MÉTODOS

4.1 Discursão geral sobre os componentes utilizados.

Será descrito aqui o passo a passo da montagem da plataforma robótica desenvolvida neste trabalho, bem como a especificação dos materiais utilizados para a montagem da mesma.

A princípio pode-se levantar que os pontos mais relevantes em relação aos componentes utilizados, são a utilização de um circuito para comunicação sem fio (para enviar comandos do computador para a plataforma), a utilização de um microcontrolador PIC 16F628A como base de controle de comandos, uma ponte H que permite a inversão de sentido de rotação de um motor CC, e pontos de alimentação em 5 V e em 6 V obtida de uma bateria de 9 V por meio de reguladores de tensão. Na tabela 1 é descrita a lista de material utilizado:

Tabela 1 – Lista de material (componentes)

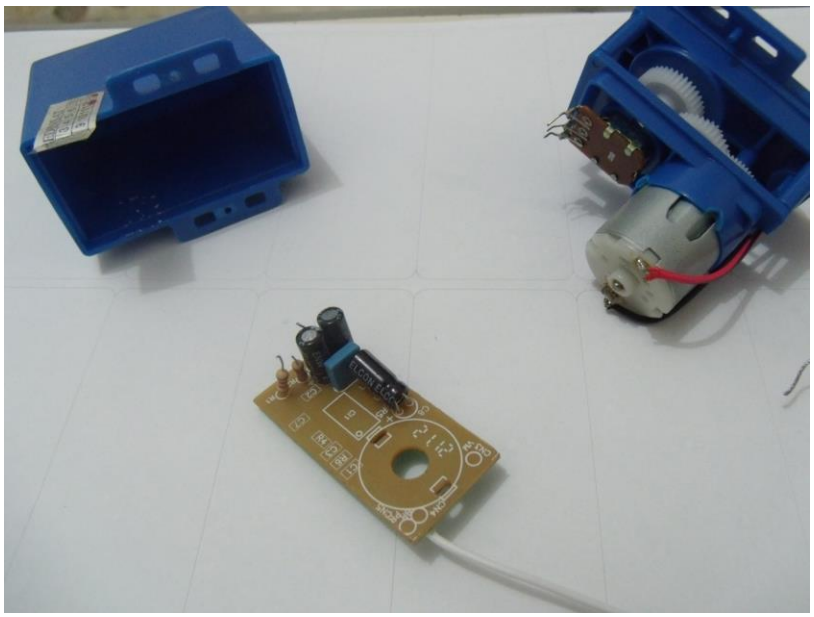
Componente	Quantidade
PIC 16F628A	1
Ponte H L298N	1
Diodo 1N4007	8
Trimpot 100k	2
Motor CC	2
Regulador de tensão 7805	1
Regulador de tensão 7806	1
Capacitor 100 nF	4
Capacitor 330 nF	2
Cristal 20 MHz	1
Capacitor 30 pF	2
Módulo Bluetooth RS232 TTL	1
Chave on/off	1
Resistor 1k	10
LED	5
Placa de fenolite 10 cm x 10 cm	2

Os motores CC utilizados foram obtidos a partir de dois servos motores do mesmo tipo que é visto na figura 4.

Figura 4 – Servo motores utilizados para a obtenção de motores CC

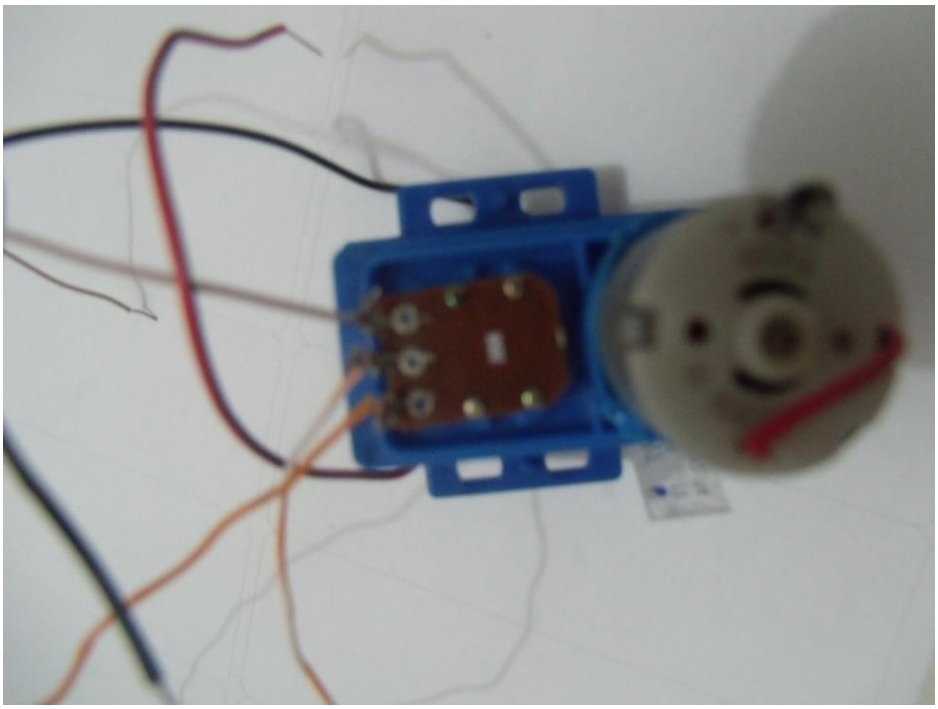
Como o controle do sentido de rotação do motor será feito pela ponte H L298N, é necessário à retirada do circuito de controle que vem acoplado ao motor, que utiliza PWM para efetuar tal trabalho. Na figura 5 pode ser visto em destaque a abertura do motor e retirada do circuito de controle do servo motor, deixando apenas o motor CC, engrenagens e um potenciômetro que será utilizado no controle da quantidade de giros do motor posteriormente.

Figura 5 – Retirada do circuito de controle do servo motor



Para finalizar as modificações feitas no motor são conectados dois fios para alimentação diretamente nos seus terminais, estes serão levados para serem conectados as saídas da ponte H no circuito. Também são conectados aos pinos do potenciômetro do conjunto motor fios que deverão ser utilizados para o controle de rotação dos motores. Desta forma, um total de cinco fios saíram de cada motor.

Figura 6 – Conectando fios de alimentação e controle de rotação ao motor CC



Os reguladores de tensão 7805 e 7806 são utilizados respectivamente para obtenção de 5 e 6 volts (necessário para a alimentação de alguns outros componentes como o Microcontrolador e a ponte H) a partir de uma bateria fonte de 9 volts, optou-se pela utilização desses dois componentes com o propósito de se diminuir o peso da plataforma final, em vista que, a utilização de pilhas resultaria em maior peso.

As duas pontes de quatro diodos cada, são necessárias devido ao efeito indutivo que os motores produzem, tendendo a fazer permanecer a passagem de uma corrente pela ponte H, mesmo quando em estado desativado na sua saída. Assim ligando os diodos em antiparalelo esse efeito é bloqueado.

O oscilador ou clock pode ser configurado de várias maneiras distintas, dependendo do uso. Esta configuração é feita via software e é aceita pelo microcontrolador durante sua gravação. O clock determinará a velocidade de operação do microcontrolador. Atualmente o PIC16F628A é distribuído com clock’s de 4MHz,

10MHz e 20MHz, sendo que para a frequência de 20 MHz deverá ser utilizado um cristal oscilador externo. Devemos escolher a freqüência máxima de operação do microcontrolador de acordo com as tarefas que o mesmo irá executar. A figura abaixo mostra uma das configurações possíveis de oscilação e na tabela o valor dos capacitores de acordo com o modo e a frequência de operação.

Figura 7 – Modos de operação para o oscilador

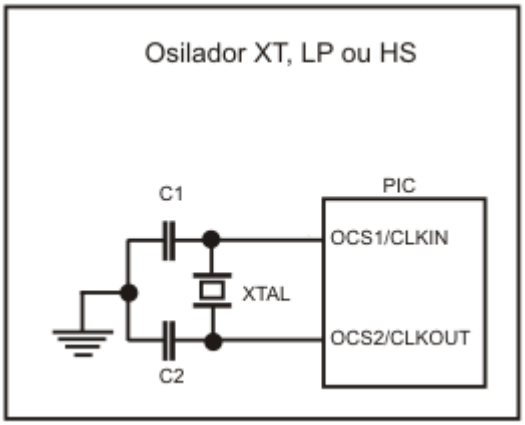


Tabela 2 – Relação modo/frequência/capacitores

Modo	Frequência	Capacitores recomendados
LP	32 KHz	68 pF – 100 pF
	200 KHz	15 pF – 33 pF
XT	2 MHz – 4 MHz	15 pF – 33 pF
HS	4 MHz – 20 MHz	15 pF – 33 pF

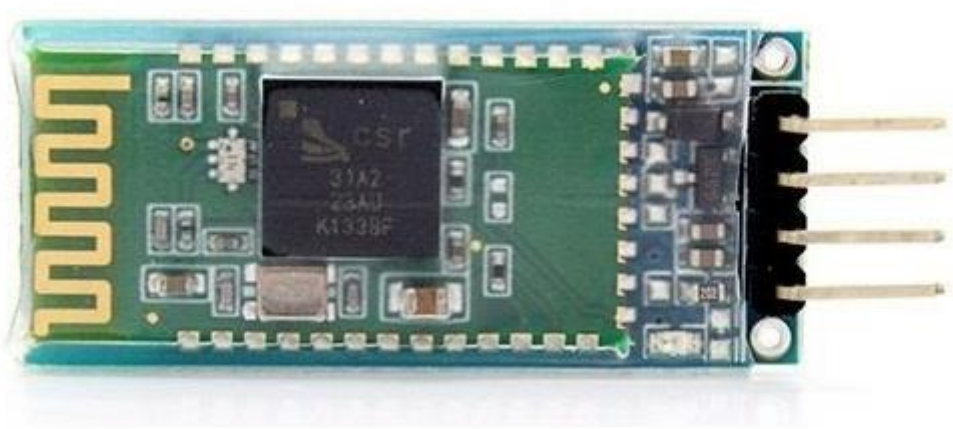
Neste projeto o cristal oscilador deve ser conectado nas portas 15 e 16 do PIC 16F628A.

4.2 Projeto e montagem.

Como já abordado anteriormente os pontos principais desse projeto estão em torno da comunicação sem fio entre o computador e a plataforma robótica, o controle feito por microcontrolador PIC 16F628A, e o chaveamento do motor através da ponte H L298N.

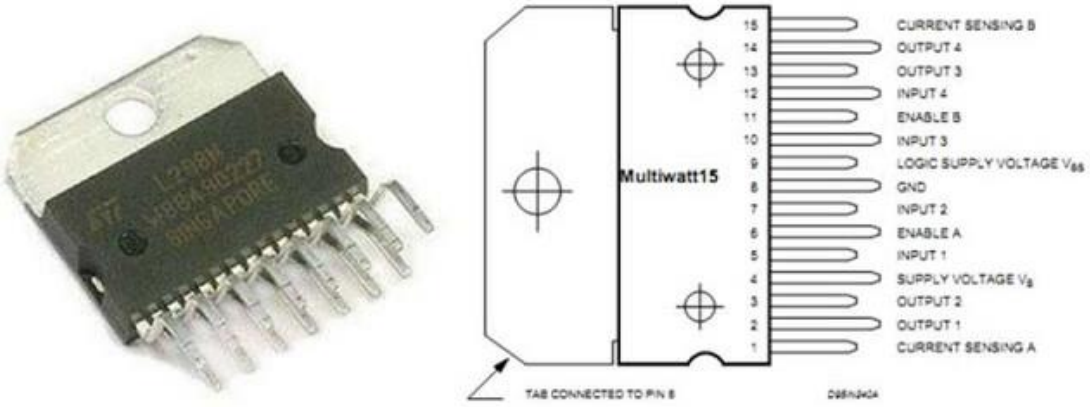
O módulo bluetooth RS232 TTL utilizado no projeto é visto na figura 8. Observando a sua configuração, podemos destacar os quatro pinos, onde de cima para baixo temos primeiro o terminal de alimentação Vcc (entre 3,6 V e 6 V), segundo o terminal de referência, e os dois últimos terminais correspondentes a comunicação com o microcontrolador, de forma a serem conectados nas portas 7 e 8 do PIC. Nesta ligação entre o módulo e o PIC não é necessária amplificação do sinal, pois a saída do módulo tem nível de tensão de 3,3 V, o que é suficiente para comunicação com o PIC.

Figura 8 – Módulo bluetooth RS232 TTL



Em relação a ponte H L298N, será conectada ao microcontrolador para receber sinais de comando que determinaram o sentido de rotação dos motores, sendo capaz de controlar dois motores ao mesmo tempo. Os pinos 5 e 7 são as entradas de sinal vindos do PIC para o controle de um motor, e os pinos 10 e 12 são as entradas de sinal para o controle do outro motor.

Figura 9 – Ponte H L298N



Também possui quatro terminais de saída, pinos 2 e 3 para um motor, e pinos 13 e 14 para o outro motor.

Nas saídas da ponte H, os motores foram ligados em paralelo com um trimpot de 100K a fim de deixar os giros dos dois motores mais próximos possíveis um do outro em relação a sua velocidade de rotação, tem-se na verdade um divisor de corrente ajustável para cada motor.

As alimentações do PIC, módulo Bluetooth e ponte H são feitas com 5 V CC, obtidos da saída do regulador 7805, a tensão de 6 V CC, obtida do regulador 7806 é utilizada chegando a ponte H para alimentação dos motores, é necessária pois o motores são alimentados em 5 V, porém há uma perda de potencial no circuito da ponte H.

Figura 10 – Regulador de tensão 7805

Figura 11 – Microcontrolador PIC 16F628A

Inicialmente os testes de montagem foram feitos em um protoboard, numa segunda etapa foram divididos os circuitos de controle e de alimentação dos motores, onde cada uma dessas foi construída em uma placa de circuito diferente, na tentativa de deixar a montagem mais modular possível. Assim, se necessária a substituição ou troca dos motores utilizados, ou mesmo do sistema de controle dos motores, a modificação será possível, apenas retirando a parte do circuito em questão.

Foi criado um circuito para impressão na placa, nas figuras 12 e 13 são mostradas as disposições de componentes e também o resultado do desenho que irá para a placa.

Feito isso, o próximo passo é gravar os caminhos de tinta na placa de fenolite, para que o cobre na placa forme as linhas de contato, como se fossem fios.

Para isto, foi utilizado o seguinte método: primeiro foi impresso o desenho do circuito em uma folha de etiquetas. Sendo que a impressão é feita na folha lisa, logo é necessária, a retirada das etiquetas (que podem ser descartadas) e utilizada a parte da folha onde elas estavam coladas.

Figura 12 – Placa com componentes de controle

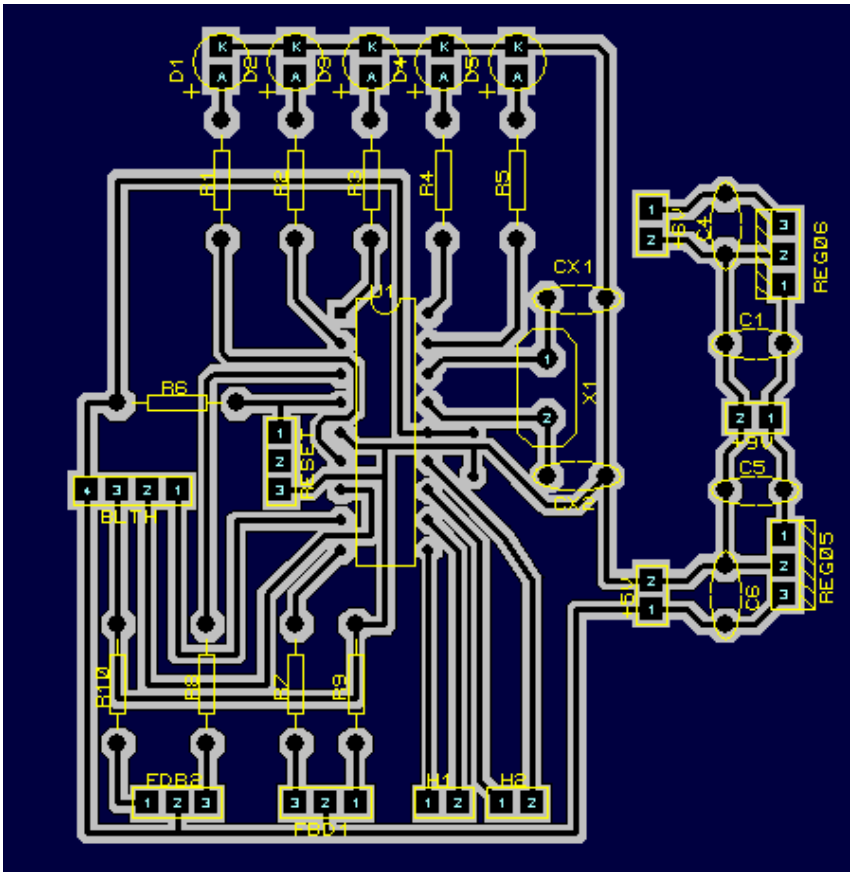


Figura 13 – Placa com componentes de controle dos motores

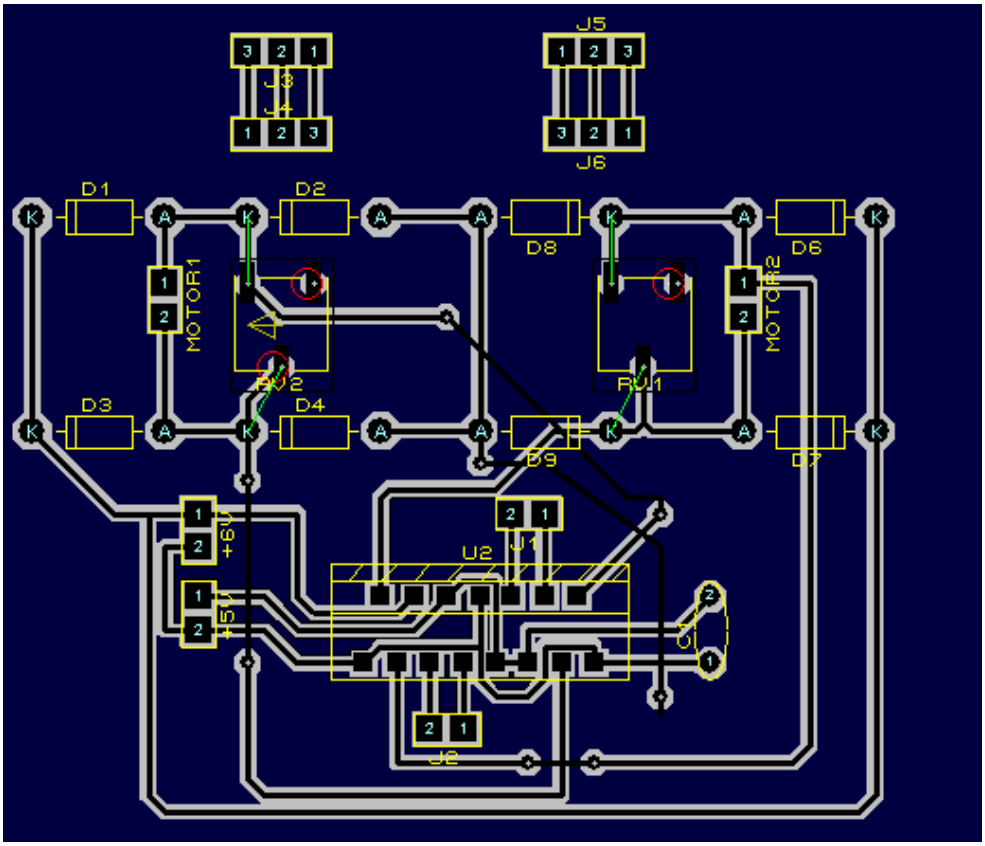
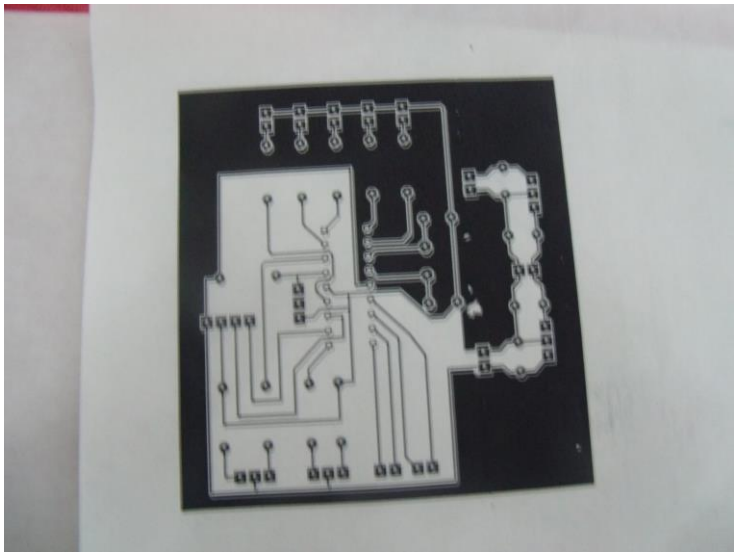


Figura 14 – Impressão de circuito em folha de etiquetas

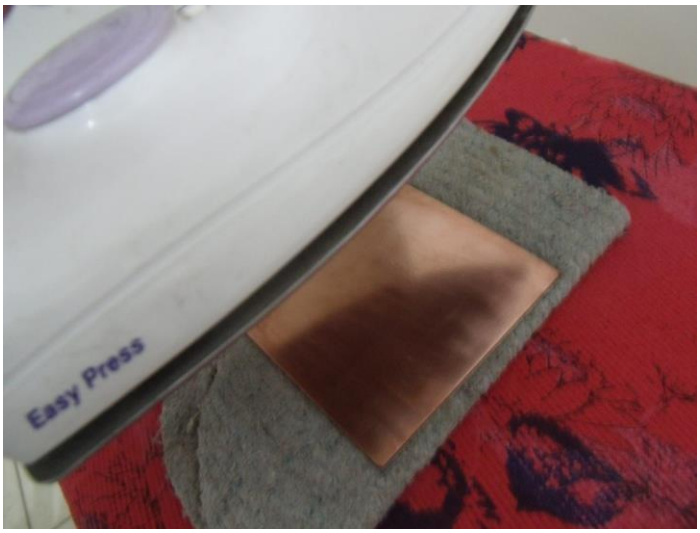


Também é necessário que a impressão seja feita com uma impressora a laser, que imprima no mínimo preto e branco, diferente disso é grande a probabilidade de a transferência do circuito, do papel para a placa não acontecer de forma devida.

Existem na internet descrição de métodos diferentes para fazer esta transferência, onde em sua grande maioria utilizam folhas de transparência, porém o método que é descrito neste trabalho, mostra-se muito mais eficiente, pois as linhas são transpostas quase todas em perfeito estado, e também gasta-se muito menos tempo para a transferência, por volta de 5 minutos.

Para a transferência da imagem do circuito para a placa propriamente dita, foi utilizado um ferro de passar roupas para aquecer a superfície cobreada da placa, o ferro pode ser colocado diretamente sobre a placa, como na figura 15.

Figura 15 – Aquecimento da superfície cobreada da placa de fenolíte



A temperatura do ferro elétrico foi utilizada no máximo, e a superfície cobreada da placa limpa, antes do processo, com lã de aço. Não existe um tempo mínimo ou máximo que o ferro deva permanecer sobre a placa, mas a placa deve ficar quente o suficiente, até não ser possível mais segurá-la com as mãos.

Depois de esquentada a placa, a folha com a impressão do circuito, é colocada com o lado da tinta que está no papel em contato direto com a face cobreada da placa de fenolite, deve-se usar um pano para pressionar o papel sobre a placa. As figuras 16 e 17 mostram os resultados deste processo.

Figura 16 – Circuito transferido para a placa de fenolite

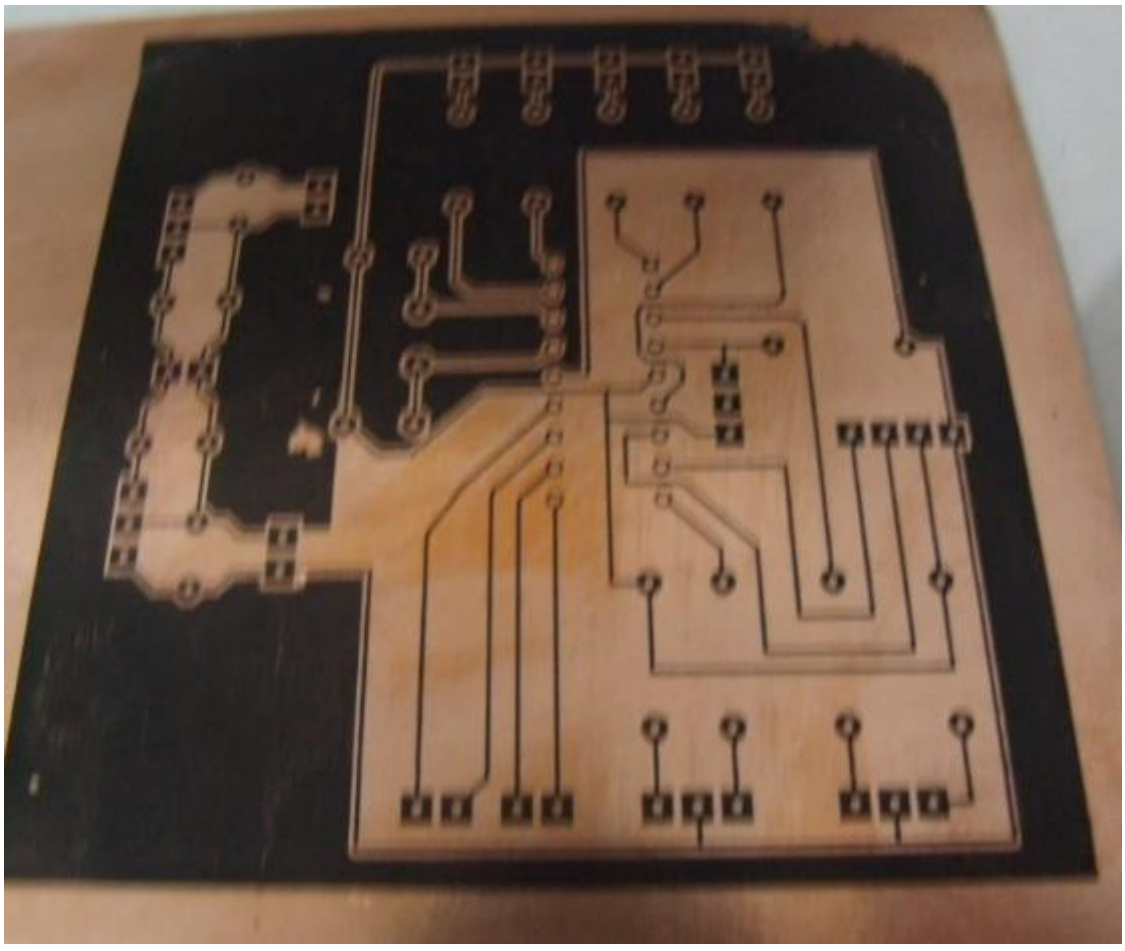
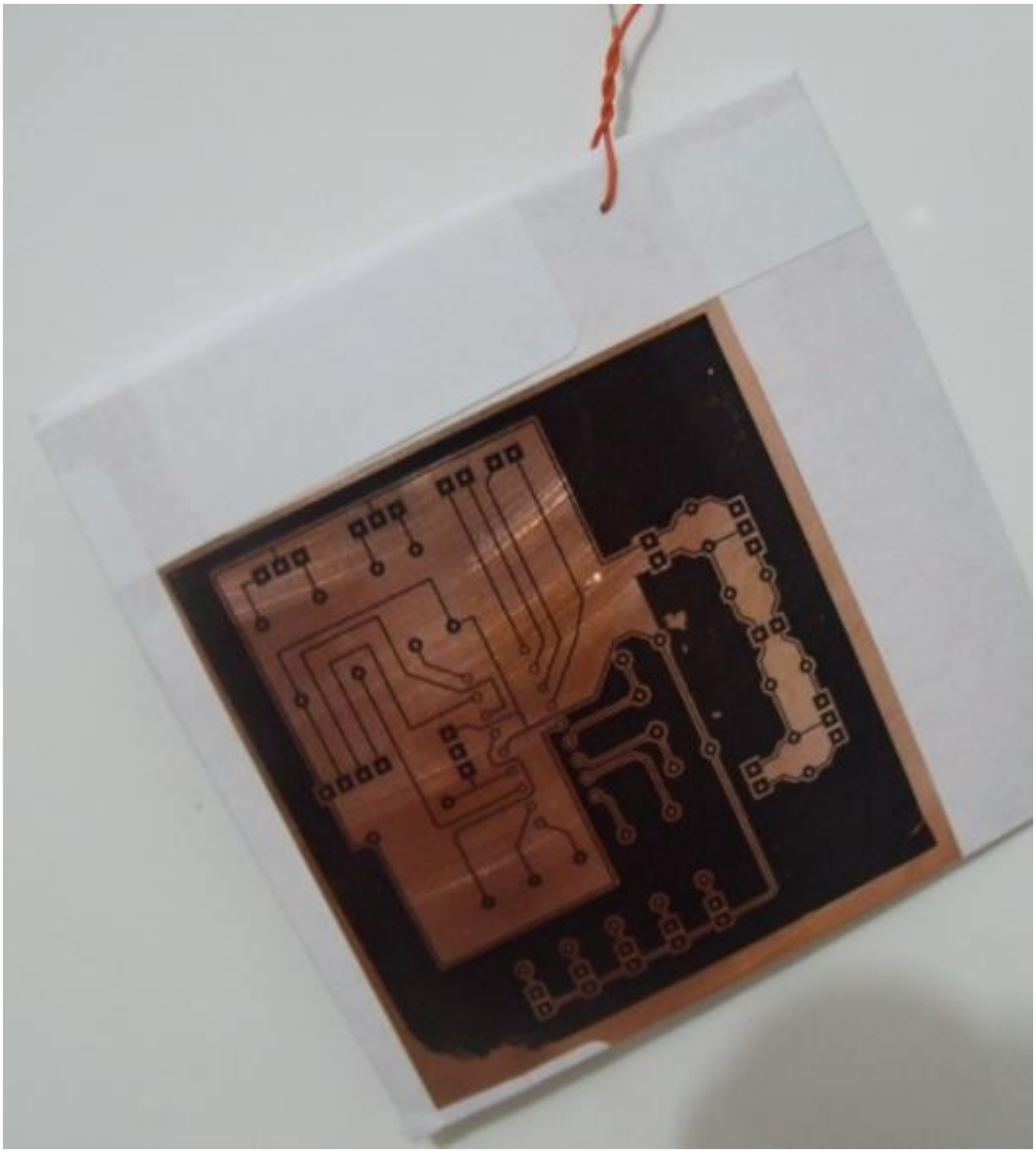


Figura 17 – Preparação da placa para corrosão do cobre em percloreto de ferro



Ao entrar em contato com o percloreto de ferro, a parte de cobre que não estiver coberta pela tinta, reagirá e será diluído, ficando na placa apenas a parte pintada. A figura 18 mostra a placa sendo mergulhada no percloreto de ferro, ao perceber que o cobre está saindo da placa e que o processo está completo, retira-se a mesma, passando-a em água corrente. Após isto a tinta pode ser retirada com uma lã de aço, já deixando os caminhos prontos na placa.

Figura 18 – Placa de fenolite em contato com percloreto de ferro

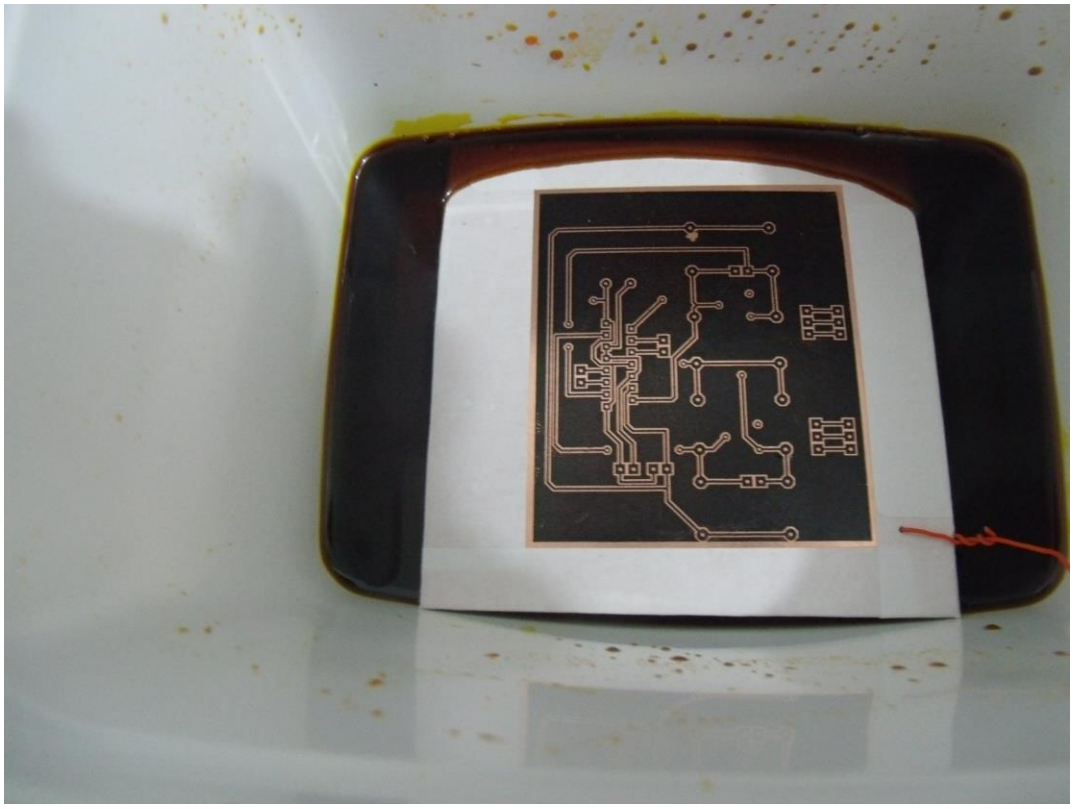
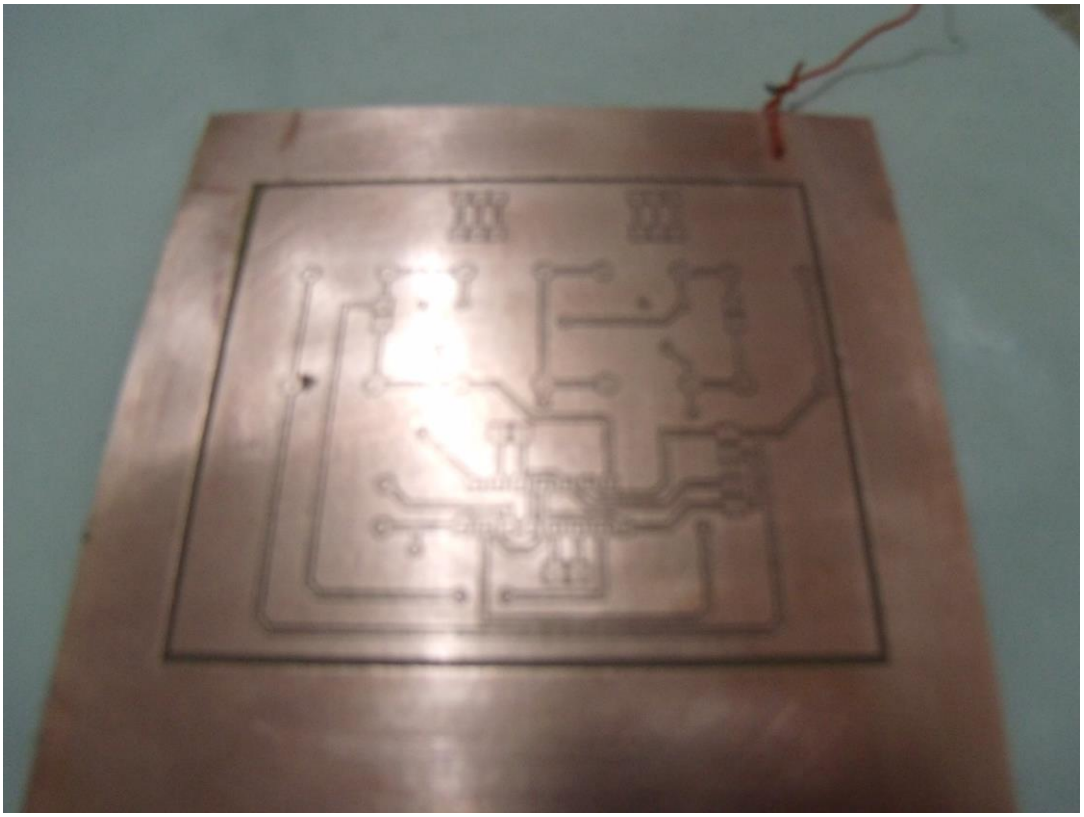
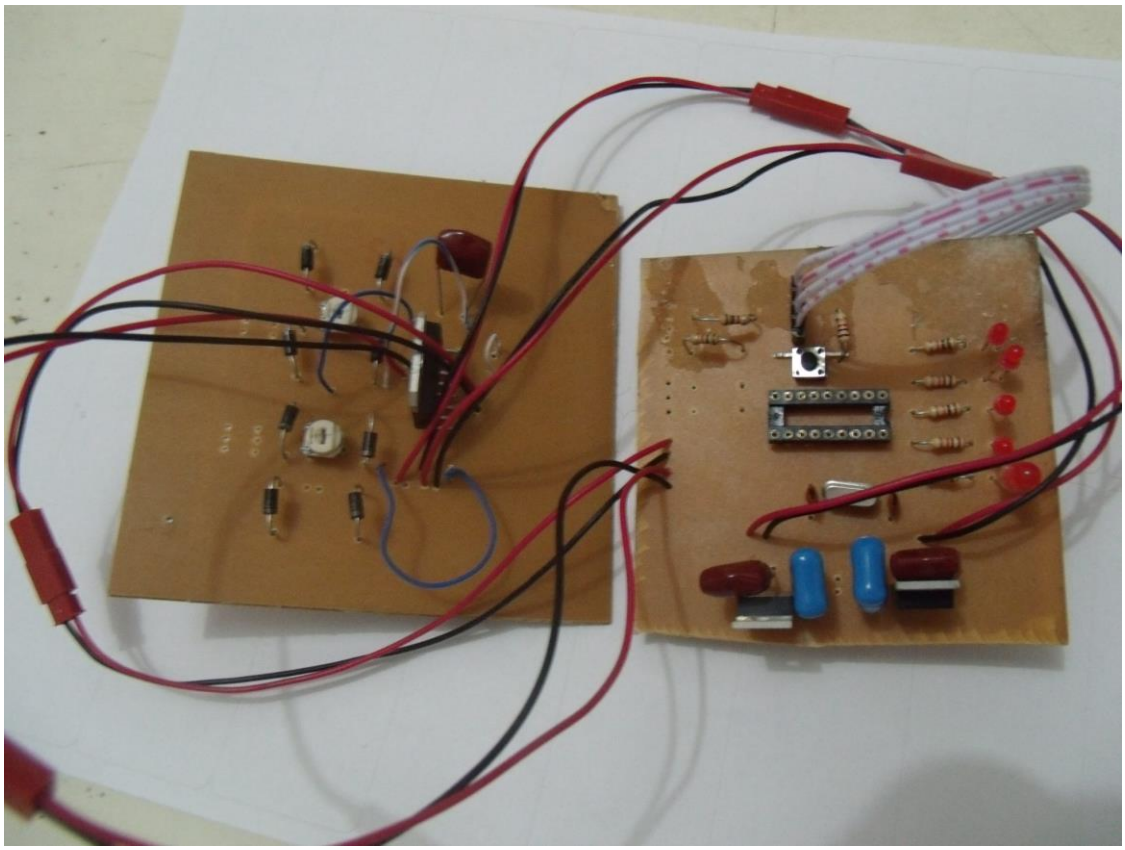


Figura 19 – Placa de fenolite após reação e limpeza da tinta



Ao final deste processo temos as placas prontas para receber os componentes e soldas. Na figura 20, mostra-se a disposição das duas placas, bem como a suas ligações feitas por cabos de encaixe.

Figura 20 – Ligação das duas placas de maneira modular.



Foi construída a plataforma para suporte dos circuitos contendo rodas, presas aos eixos dos motores, para o seu deslocamento. É importante dizer que a mesma foi feita utilizando material reciclável, ou materiais que possam ser obtidas facilmente em qualquer residência. Inicialmente foram utilizadas latas de refrigerante, mas pode ser feito utilizando papelão.

5 RESULTADOS E DISCUSSÃO

Aproveitando a lista de material descrita na tabela 1, vamos elaborar uma tabela de custos, para ter uma ideia do custo médio para a construção desta plataforma robótica.

Tabela 3 – Lista de material e custo

Componente	Quantidade	Preço Unitário (R$)	Preço (R$)
PIC 16F628A	1	10,00	10,00
Ponte H L298N	1	8,00	8,00
Diodo 1N4007	8	0,15	1,20
Trimpot 100k	2	0,30	0,60
Motor CC	2	25,00	50,00
Regulador de tensão 7805	1	0,75	0,75
Regulador de tensão 7806	1	0,75	0,75
Capacitor 100 nF	4	0,50	2,00
Capacitor 330 nF	2	0,50	1,00
Cristal 20 MHz	1	0,50	0,50
Capacitor 30 pF	2	0,20	0,40
Módulo Bluetooth To Rs232 Ttl	1	16,00	16,00
Chave on/off	1	2,00	2,00
Resistor 1k	10	0,15	0,15
LED	5	0,50	2,50
Placa de fenolite 10x10	2	3,50	7,00
		Total (R$)	102,85

Comparando com custos de kits utilizados na robótica educacional que são vendidos no mercado, temos que a plataforma desenvolvida tem um custo bastante reduzido em relação às outras, porém é importante dizer também, que os demais kits possuem maior quantidade de funcionalidades já que não foram inseridos sensores que possibilitam maior interação com o meio.

Mediante alguns problemas ocorridos com a ponte H e com as placas de circuito impresso utilizados, a apresentação do trabalho acabou sendo realizada apenas com o circuito funcionando na protoboard. A gravação do programa no PIC 16F628A foi feita a partir de uma placa de circuito para gravação de microcontroladores. O programa utilizado encontrasse no Anexo C.

O controle da plataforma é feito por um programa de computador desenvolvido no trabalho "DESENVOLVIMENTO DE UMA INTERFACE GRÁFICA PARA APLICAÇÃO EM ROBÓTICA EDUCACIONAL DE BAIXO CUSTO", referenciado neste trabalho e servindo de complemento para o mesmo.

Muitas melhorias podem ser feitas neste trabalho, a continuação do mesmo pode seguir no caminho de primeiramente corrigir os problemas com as placas de circuito impresso, estudar a utilização de um motor mais barato é que forneça maior torque e também a substituição da ponte H por um circuito com base em transistores que também ajudará a reduzir os custos do projeto final.

Tabela 4 – Custo de kits para robótica educacional

Kit Robótico	Fabricante	Preço (R$)
Lego Mindstorms 2.0	LEGO	1000,00
Kit robô KIROBO	ELEKIT	256,00
Kit para robótica	MicroKids	230,00
Lego Mindstorms EV3	LEGO	1700,00

6 CONCLUSÃO

A plataforma robótica desenvolvida neste trabalho apresentou resultado satisfatório em relação ao seu reduzido custo, mostrando estar em um patamar de valor para construção menor do que o valor de compra de um kit comercial. Porém melhorias ainda podem ser feitas, tanto no quesito funcionalidades quanto no quesito custos.

Um dos componentes mais caros da construção são justamente os motores, que são os atuadores dentro desse sistema de controle, logo um ponto a se estudar seria exatamente a troca dos motores por um mais barato, possivelmente um motor CC, que não necessite de modificação.

O acréscimo de sensores não é realidade muito distante, com a plataforma que se tem hoje, facilmente poderia ser acrescentado um sensor de toque no lugar de uma das portas que estão alimentando os leds. Bastaria utilizar um botão on/off que ativasse a porta do pino quando algo encostasse e o fechasse.

Dentro deste contexto e dos resultados obtidos, podemos dizer que a robótica pode ser utilizada de maneira mais acessível dentro das atividades de ensino para crianças, mesmo em escolas da rede pública quando recorremos a alternativas de projeto e montagem diferentes dos kits comerciais.

REFERÊNCIAS

Silvana do Rocio Zilli, A Robótica Educacional no Ensino Fundamental: Perspectivas e Prática, Dissertação de Mestrado, Universidade Federal de Santa Catarina, Florianópolis – SC, 2004.

GONÇALVES, P. C. Protótipo de um Robô móvel de baixo custo para uso educacional. Dissertação (mestrado) - Universidade Estadual de Maringá. Programa de Pós-graduação em Ciência da Computação, 2007.

MARTIN, F. Cybernetics, and Programmable Turtles. 1988. 87 p. Thesis (Master of Science in Mechanical Engineering) - Departament of Mechanical Egineering, Massachusetts Institute of Technology (MIT), Cambridge, 1988.

AZEVEDO, S.; AGLAÉ, A.; PITTA, R. Minicurso: Introdução a Robótica Educacional. Disponível em: <http://www.sbpcnet.org.br/livro/62ra/minicursos/MC%20Samuel%20Azevedo.pdf>. Acesso em: 30 de maio de 2013.

FILHO, C. M. de O. L.; FREIRE, R. C. S.; Desenvolvimento de uma Plataforma Didática de Baixo Custo para o Ensino de Robótica. UFCG. Disponível em: <http://limcserver.dee.ufcg.edu.br/semetro/www/pdf/52379_1.pdf >. Acesso em: 1 de junho de 2013.

Fernando Pazos, Automação de Sistemas & Robótica, Axel Books, Rio de Janeiro, 2002.

COSTA, V. de P. DESENVOLVIMENTO DE UMA INTERFACE GRÁFICA PARA APLICAÇÃO EM ROBÓTICA EDUCACIONAL DE BAIXO CUSTO. Trabalho de conclusão de curso, UFERSA. Mossoró – RN, 2013.

ANEXO

Anexo A – Datasheet do microcontrolador.

MICROCHIP

PIC16F627A/628A/648A

18-pin Flash-Based, 8-Bit CMOS Microcontrollers with nanoWatt Technology

High-Performance RISC CPU:

- Operating speeds from DC – 20 MHz
- Interrupt capability
- 8-level deep hardware stack
- Direct, Indirect and Relative Addressing modes
- 35 single-word instructions:
  - All instructions single cycle except branches

Special Microcontroller Features:

- Internal and external oscillator options:
  - Precision internal 4 MHz oscillator factory calibrated to ±1%
  - Low-power internal 48 kHz oscillator
  - External Oscillator support for crystals and resonators
- Power-saving Sleep mode
- Programmable weak pull-ups on PORTB
- Multiplexed Master Clear/Input-pin
- Watchdog Timer with independent oscillator for reliable operation
- Low-voltage programming
- In-Circuit Serial Programming™ (via two pins)
- Programmable code protection
- Brown-out Reset
- Power-on Reset
- Power-up Timer and Oscillator Start-up Timer
- Wide operating voltage range (2.0-5.5V)
- Industrial and extended temperature range
- High-Endurance Flash/EEPROM cell:
  - 100,000 write Flash endurance
  - 1,000,000 write EEPROM endurance
  - 40 year data retention

Low-Power Features:

- Standby Current:
  - 100 nA @ 2.0V, typical
- Operating Current:
  - 12 µA @ 32 kHz, 2.0V, typical
  - 120 µA @ 1 MHz, 2.0V, typical
- Watchdog Timer Current:
  - 1 µA @ 2.0V, typical
- Timer1 Oscillator Current:
  - 1.2 µA @ 32 kHz, 2.0V, typical
- Dual-speed Internal Oscillator:
  - Run-time selectable between 4 MHz and 48 kHz
  - 4 µs wake-up from Sleep, 3.0V, typical

Peripheral Features:

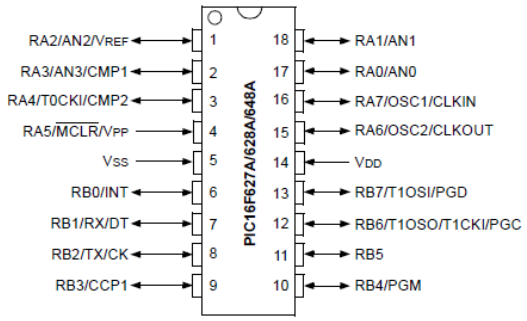
- 16 I/O pins with individual direction control
- High current sink/source for direct LED drive
- Analog comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (V_{REF}) module
  - Selectable internal or external reference
  - Comparator outputs are externally accessible
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler
- Timer1: 16-bit timer/counter with external crystal/clock capability
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Capture, Compare, PWM module:
  - 16-bit Capture/Compare
  - 10-bit PWM
- Addressable Universal Synchronous/Asynchronous Receiver/Transmitter USART/SCI

Device	Program Memory	Data Memory		I/O	CCP (PWM)	USART	Comparators	Timers 8/16-bit
	Flash (words)	SRAM (bytes)	EEPROM (bytes)					
PIC16F627A	1024	224	128	16	1	Y	2	2/1
PIC16F628A	2048	224	128	16	1	Y	2	2/1
PIC16F648A	4096	256	256	16	1	Y	2	2/1

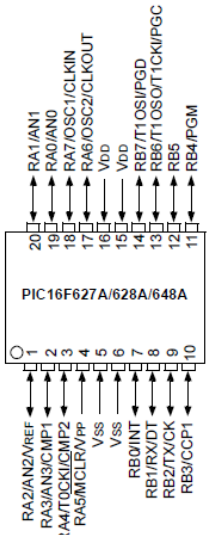
PIC16F627A/628A/648A

Pin Diagrams

PDIP, SOIC



SSOP



28-Pin QFN

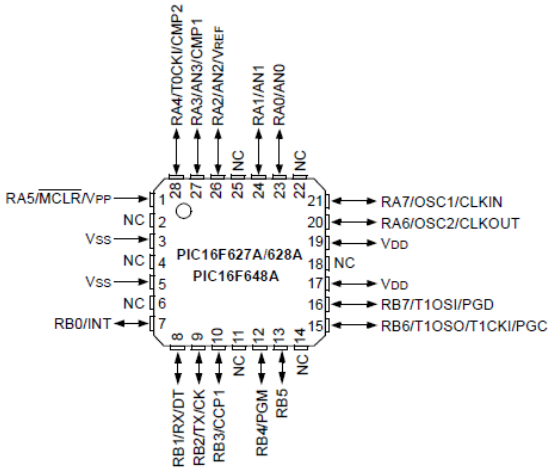


TABLE 3-2: PIC16F627A/628A/648A PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
RA0/AN0	RA0	ST	CMOS	Bidirectional I/O port
	AN0	AN	—	Analog comparator input
RA1/AN1	RA1	ST	CMOS	Bidirectional I/O port
	AN1	AN	—	Analog comparator input
RA2/AN2/VREF	RA2	ST	CMOS	Bidirectional I/O port
	AN2	AN	—	Analog comparator input
	VREF	—	AN	VREF output
RA3/AN3/CMP1	RA3	ST	CMOS	Bidirectional I/O port
	AN3	AN	—	Analog comparator input
	CMP1	—	CMOS	Comparator 1 output
RA4/T0CKI/CMP2	RA4	ST	OD	Bidirectional I/O port
	T0CKI	ST	—	Timer0 clock input
	CMP2	—	OD	Comparator 2 output
RA5/$\overline{\text{MCLR}}$/VPP	RA5	ST	—	Input port
	$\overline{\text{MCLR}}$	ST	—	Master clear. When configured as $\overline{\text{MCLR}}$, this pin is an active low Reset to the device. Voltage on $\overline{\text{MCLR}}$/VPP must not exceed VDD during normal device operation.
	VPP	—	—	Programming voltage input
RA6/OSC2/CLKOUT	RA6	ST	CMOS	Bidirectional I/O port
	OSC2	—	XTAL	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
	CLKOUT	—	CMOS	In RC/INTOSC mode, OSC2 pin can output CLKOUT, which has 1/4 the frequency of OSC1.
RA7/OSC1/CLKIN	RA7	ST	CMOS	Bidirectional I/O port
	OSC1	XTAL	—	Oscillator crystal input
	CLKIN	ST	—	External clock source input. RC biasing pin.
RB0/INT	RB0	TTL	CMOS	Bidirectional I/O port. Can be software programmed for internal weak pull-up.
	INT	ST	—	External interrupt
RB1/RX/DT	RB1	TTL	CMOS	Bidirectional I/O port. Can be software programmed for internal weak pull-up.
	RX	ST	—	USART receive pin
	DT	ST	CMOS	Synchronous data I/O
RB2/TX/CK	RB2	TTL	CMOS	Bidirectional I/O port. Can be software programmed for internal weak pull-up.
	TX	—	CMOS	USART transmit pin
	CK	ST	CMOS	Synchronous clock I/O
RB3/CCP1	RB3	TTL	CMOS	Bidirectional I/O port. Can be software programmed for internal weak pull-up.
	CCP1	ST	CMOS	Capture/Compare/PWM I/O

Legend: O = Output, — = Not used, TTL = TTL Input, CMOS = CMOS Output, I = Input, OD = Open Drain Output, P = Power, ST = Schmitt Trigger Input, AN = Analog

TABLE 3-2: PIC16F627A/628A/648A PINOUT DESCRIPTION (CONTINUED)

Name	Function	Input Type	Output Type	Description
RB4/PGM	RB4	TTL	CMOS	Bidirectional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	PGM	ST	—	Low-voltage programming input pin. When low-voltage programming is enabled, the interrupt-on-pin change and weak pull-up resistor are disabled.
RB5	RB5	TTL	CMOS	Bidirectional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
RB6/T1OSO/T1CKI/PGC	RB6	TTL	CMOS	Bidirectional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	T1OSO	—	XTAL	Timer1 oscillator output
	T1CKI	ST	—	Timer1 clock input
	PGC	ST	—	ICSP™ programming clock
RB7/T1OSI/PGD	RB7	TTL	CMOS	Bidirectional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	T1OSI	XTAL	—	Timer1 oscillator input
	PGD	ST	CMOS	ICSP data I/O
VSS	VSS	Power	—	Ground reference for logic and I/O pins
VDD	VDD	Power	—	Positive supply for logic and I/O pins

Legend: O = Output, — = Not used, TTL = TTL Input, CMOS = CMOS Output, I = Input, OD = Open Drain Output, P = Power, ST = Schmitt Trigger Input, AN = Analog

11.0 VOLTAGE REFERENCE MODULE

The Voltage Reference module consists of a 16-tap resistor ladder network that provides a selectable voltage reference. The resistor ladder is segmented to provide two ranges of VREF values and has a power-down function to conserve power when the reference is not being used. The VRCON register controls the operation of the reference as shown in Figure 11-1. The block diagram is given in Figure 11-1.

11.1 Voltage Reference Configuration

The Voltage Reference module can output 16 distinct voltage levels for each range.

The equations used to calculate the output of the Voltage Reference module are as follows:

if VRR = 1:

$$V_{REF} = \frac{V_R<3:0>}{24} \times V_{DD}$$

if VRR = 0:

$$V_{REF} = \left(V_{DD} \times \frac{1}{4}\right) + \frac{V_R<3:0>}{32} \times V_{DD}$$

The setting time of the Voltage Reference module must be considered when changing the VREF output (Table 17-3). Example 11-1 demonstrates how voltage reference is configured for an output voltage of 1.25V with VDD = 5.0V.

REGISTER 11-1: VRCON – VOLTAGE REFERENCE CONTROL REGISTER (ADDRESS: 9Fh)

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
VREN	VROE	VRR	—	VR3	VR2	VR1	VR0
bit 7							bit 0

bit 7 **VREN**: VREF Enable bit
1 = VREF circuit powered on
0 = VREF circuit powered down, no IDD drain

bit 6 **VROE**: VREF Output Enable bit
1 = VREF is output on RA2 pin
0 = VREF is disconnected from RA2 pin

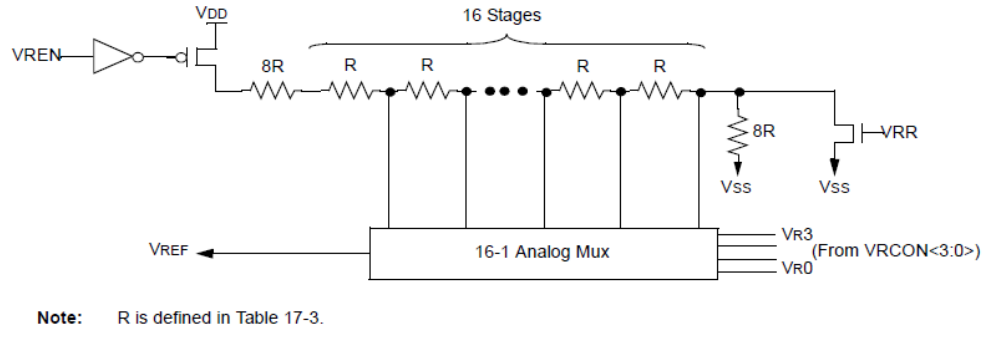
bit 5 **VRR**: VREF Range Selection bit
1 = Low range
0 = High range

bit 4 **Unimplemented**: Read as '0'

bit 3-0 **VR<3:0>**: VREF Value Selection bits 0 ≤ VR <3:0> ≤ 15
When VRR = 1: VREF = (VR<3:0>/ 24) * VDD
When VRR = 0: VREF = 1/4 * VDD + (VR<3:0>/ 32) * VDD

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

FIGURE 11-1: VOLTAGE REFERENCE BLOCK DIAGRAM



Note: R is defined in Table 17-3.

EXAMPLE 11-1: VOLTAGE REFERENCE CONFIGURATION

```
MOVLW   0x02        ;4 Inputs Muxed
MOVWF   CMCON       ;to 2 comps.
BSF     STATUS,RP0  ;go to Bank 1
MOVLW   0x07        ;RA3-RA0 are
MOVWF   TRISA       ;outputs
MOVLW   0xA6        ;enable VREF
MOVWF   VRCON       ;low range set VR<3:0>=6
BCF     STATUS,RP0  ;go to Bank 0
CALL    DELAY10     ;10µs delay
```

11.2 Voltage Reference Accuracy/Error

The full range of VSS to VDD cannot be realized due to the construction of the module. The transistors on the top and bottom of the resistor ladder network (Figure 11-1) keep VREF from approaching VSS or VDD. The Voltage Reference module is VDD derived and therefore, the VREF output changes with fluctuations in VDD. The tested absolute accuracy of the Voltage Reference module can be found in Table 17-3.

11.3 Operation During Sleep

When the device wakes up from Sleep through an interrupt or a Watchdog Timer time out, the contents of the VRCON register are not affected. To minimize current consumption in Sleep mode, the Voltage Reference module should be disabled.

11.4 Effects of a Reset

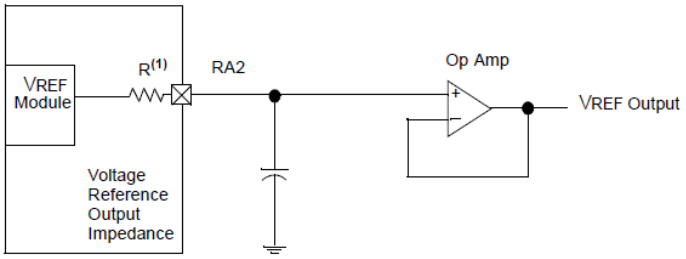
A device Reset disables the Voltage Reference module by clearing bit VREN (VRCON<7>). This Reset also disconnects the reference from the RA2 pin by clearing bit VROE (VRCON<6>) and selects the high voltage range by clearing bit VRR (VRCON<5>). The VREF value select bits, VRCON<3:0>, are also cleared.

11.5 Connection Considerations

The Voltage Reference module operates independently of the Comparator module. The output of the reference generator may be connected to the RA2 pin if the TRISA<2> bit is set and the VROE bit, VRCON<6>, is set. Enabling the Voltage Reference module output onto the RA2 pin with an input signal present will increase current consumption. Connecting RA2 as a digital output with VREF enabled will also increase current consumption.

The RA2 pin can be used as a simple D/A output with limited drive capability. Due to the limited drive capability, a buffer must be used in conjunction with the Voltage Reference module output for external connections to VREF. Figure 11-2 shows an example buffering technique.

FIGURE 11-2: VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE



Note 1: R is dependent upon the voltage reference configuration VRCON<3:0> and VRCON<5>.

TABLE 11-1: REGISTERS ASSOCIATED WITH VOLTAGE REFERENCE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value On POR	Value On All Other Resets
9Fh	VRCON	VREN	VROE	VRR	—	VR3	VR2	VR1	VR0	000- 0000	000- 0000
1Fh	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0000	0000 0000
85h	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1111 1111	1111 1111

Legend: - = Unimplemented, read as '0'.

12.0 UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART) MODULE

The Universal Synchronous Asynchronous Receiver Transmitter (USART) is also known as a Serial Communications Interface (SCI). The USART can be configured as a full-duplex asynchronous system that can communicate with peripheral devices such as CRT terminals and personal computers, or it can be configured as a half-duplex synchronous system that can communicate with peripheral devices such as A/D or D/A integrated circuits, Serial EEPROMs, etc.

The USART can be configured in the following modes:

- Asynchronous (full-duplex)
- Synchronous – Master (half-duplex)
- Synchronous – Slave (half-duplex)

Bit SPEN (RCSTA<7>) and bits TRISB<2:1> have to be set in order to configure pins RB2/TX/CK and RB1/RX/DT as the Universal Synchronous Asynchronous Receiver Transmitter.

Register 12-1 shows the Transmit Status and Control Register (TXSTA) and Register 12-2 shows the Receive Status and Control Register (RCSTA).

REGISTER 12-1: TXSTA – TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS: 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC**: Clock Source Select bit
Asynchronous mode
Don't care
Synchronous mode
1 = Master mode (Clock generated internally from BRG)
0 = Slave mode (Clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit(1)
1 = Transmit enabled
0 = Transmit disabled

bit 4 **SYNC**: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **Unimplemented**: Read as '0'

bit 2 **BRGH**: High Baud Rate Select bit
Asynchronous mode
1 = High speed
0 = Low speed
Synchronous mode
Unused in this mode

bit 1 **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D**: 9th bit of transmit data. Can be parity bit.

Note 1: SREN/CREN overrides TXEN in SYNC mode.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 12-2: RCSTA – RECEIVE STATUS AND CONTROL REGISTER (ADDRESS: 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN**: Serial Port Enable bit
(Configures RB1/RX/DT and RB2/TX/CK pins as serial port pins when bits TRISB<2:1> are set)
1 = Serial port enabled
0 = Serial port disabled

bit 6 **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode:
Don't care
Synchronous mode - master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode - slave:
Unused in this mode

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive
0 = Disables continuous receive
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADEN**: Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enable interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit
Asynchronous mode 8-bit (RX9 = 0):
Unused in this mode
Synchronous mode
Unused in this mode

bit 2 **FERR**: Framing Error bit
1 = Framing error (Can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR**: Overrun Error bit
1 = Overrun error (Can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D**: 9th bit of received data (Can be parity bit)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

12.1 USART Baud Rate Generator (BRG)

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit baud rate generator. The SPBRG register controls the period of a free running 8-bit timer. In Asynchronous mode, bit BRGH (TXSTA<2>) also controls the baud rate. In Synchronous mode, bit BRGH is ignored. Table 12-1 shows the formula for computation of the baud rate for different USART modes, which only apply in Master mode (internal clock).

Given the desired baud rate and Fosc, the nearest integer value for the SPBRG register can be calculated using the formula in Table 12-1. From this, the error in baud rate can be determined.

Example 12-1 shows the calculation of the baud rate error for the following conditions:

Fosc = 16 MHz

Desired Baud Rate = 9600

BRGH = 0

SYNC = 0

EQUATION 12-1: CALCULATING BAUD RATE ERROR

$$Desired\ Baud\ Rate = \frac{Fosc}{64(x+1)}$$

$$9600 = \frac{16000000}{64(x+1)}$$

$$x = 25.042$$

$$Calculated\ Baud\ Rate = \frac{16000000}{64(25+1)} = 9615$$

$$Error = \frac{(Calculated\ Baud\ Rate - Desired\ Baud\ Rate)}{Desired\ Baud\ Rate}$$

$$= \frac{9615 - 9600}{9600} = 0.16\%$$

It may be advantageous to use the high baud rate (BRGH = 1) even for slower baud clocks. This is because the Fosc/(16(X + 1)) equation can reduce the baud rate error in some cases.

Writing a new value to the SPBRG register causes the BRG timer to be reset (or cleared) and ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

The data on the RB1/RX/DT pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

TABLE 12-1: BAUD RATE FORMULA

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = Fosc/(64(X+1))	Baud Rate = Fosc/(16(X+1))
1	(Synchronous) Baud Rate = Fosc/(4(X+1))	NA

Legend: X = value in SPBRG (0 to 255)

TABLE 12-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other Resets
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
18h	RCSTA	SPEN	RX9	SREN	CREN	ADEN	FERR	OERR	RX9D	0000 000x	0000 000x
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented read as '0'. Shaded cells are not used for the BRG.

TABLE 12-3: BAUD RATES FOR SYNCHRONOUS MODE

BAUD RATE (K)	Fosc = 20 MHz		SPBRG value (decimal)	16 MHz		SPBRG value (decimal)	10 MHz		SPBRG value (decimal)
	KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR	
0.3	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	NA	—	—
2.4	NA	—	—	NA	—	—	NA	—	—
9.6	NA	—	—	NA	—	—	9.766	+1.73%	255
19.2	19.53	+1.73%	255	19.23	+0.16%	207	19.23	+0.16%	129
76.8	76.92	+0.16%	64	76.92	+0.16%	51	75.76	-1.36%	32
96	96.15	+0.16%	51	95.24	-0.79%	41	96.15	+0.16%	25
300	294.1	-1.96	16	307.69	+2.56%	12	312.5	+4.17%	7
500	500	0	9	500	0	7	500	0	4
HIGH	5000	—	0	4000	—	0	2500	—	0
LOW	19.53	—	255	15.625	—	255	9.766	—	255

BAUD RATE (K)	Fosc = 7.15909 MHz		SPBRG value (decimal)	5.0688 MHz		SPBRG value (decimal)	4 MHz		SPBRG value (decimal)
	KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR	
0.3	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	NA	—	—
2.4	NA	—	—	NA	—	—	NA	—	—
9.6	9.622	+0.23%	185	9.6	0	131	9.615	+0.16%	103
19.2	19.24	+0.23%	92	19.2	0	65	19.231	+0.16%	51
76.8	77.82	+1.32	22	79.2	+3.13%	15	75.923	+0.16%	12
96	94.20	-1.88	18	97.48	+1.54%	12	1000	+4.17%	9
300	298.3	-0.57	5	316.8	5.60%	3	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	1789.8	—	0	1267	—	0	100	—	0
LOW	6.991	—	255	4.950	—	255	3.906	—	255

BAUD RATE (K)	Fosc = 3.579545 MHz		SPBRG value (decimal)	1 MHz		SPBRG value (decimal)	32.768 kHz		SPBRG value (decimal)
	KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR	
0.3	NA	—	—	NA	—	—	0.303	+1.14%	26
1.2	NA	—	—	1.202	+0.16%	207	1.170	-2.48%	6
2.4	NA	—	—	2.404	+0.16%	103	NA	—	—
9.6	9.622	+0.23%	92	9.615	+0.16%	25	NA	—	—
19.2	19.04	-0.83%	46	19.24	+0.16%	12	NA	—	—
76.8	74.57	-2.90%	11	83.34	+8.51%	2	NA	—	—
96	99.43	+3.57%	8	NA	—	—	NA	—	—
300	298.3	0.57%	2	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	894.9	—	0	250	—	0	8.192	—	0
LOW	3.496	—	255	0.9766	—	255	0.032	—	255

TABLE 12-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)

BAUD RATE (K)	Fosc = 20 MHz		SPBRG value (decimal)	16 MHz		SPBRG value (decimal)	10 MHz		SPBRG value (decimal)
	KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR	
0.3	NA	—	—	NA	—	—	NA	—	—
1.2	1.221	+1.73%	255	1.202	+0.16%	207	1.202	+0.16%	129
2.4	2.404	+0.16%	129	2.404	+0.16%	103	2.404	+0.16%	64
9.6	9.469	-1.36%	32	9.615	+0.16%	25	9.766	+1.73%	15
19.2	19.53	+1.73%	15	19.23	+0.16%	12	19.53	+1.73V	7
76.8	78.13	+1.73%	3	83.33	+8.51%	2	78.13	+1.73%	1
96	104.2	+8.51%	2	NA	—	—	NA	—	—
300	312.5	+4.17%	0	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	312.5	—	0	250	—	0	156.3	—	0
LOW	1.221	—	255	0.977	—	255	0.6104	—	255

BAUD RATE (K)	Fosc = 7.15909 MHz		SPBRG value (decimal)	5.0688 MHz		SPBRG value (decimal)	4 MHz		SPBRG value (decimal)
	KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR	
0.3	NA	—	—	0.31	+3.13%	255	0.3005	-0.17%	207
1.2	1.203	+0.23%	92	1.2	0	65	1.202	+1.67%	51
2.4	2.380	-0.83%	46	2.4	0	32	2.404	+1.67%	25
9.6	9.322	-2.90%	11	9.9	+3.13%	7	NA	—	—
19.2	18.64	-2.90%	5	19.8	+3.13%	3	NA	—	—
76.8	NA	—	—	79.2	+3.13%	0	NA	—	—
96	NA	—	—	NA	—	—	NA	—	—
300	NA	—	—	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	111.9	—	0	79.2	—	0	62.500	—	0
LOW	0.437	—	255	0.3094	—	255	3.906	—	255

BAUD RATE (K)	Fosc = 3.579545 MHz		SPBRG value (decimal)	1 MHz		SPBRG value (decimal)	32.768 kHz		SPBRG value (decimal)
	KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR	
0.3	0.301	+0.23%	185	0.300	+0.16%	51	0.256	-14.67%	1
1.2	1.190	-0.83%	46	1.202	+0.16%	12	NA	—	—
2.4	2.432	+1.32%	22	2.232	-6.99%	6	NA	—	—
9.6	9.322	-2.90%	5	NA	—	—	NA	—	—
19.2	18.64	-2.90%	2	NA	—	—	NA	—	—
76.8	NA	—	—	NA	—	—	NA	—	—
96	NA	—	—	NA	—	—	NA	—	—
300	NA	—	—	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	55.93	—	0	15.63	—	0	0.512	—	0
LOW	0.2185	—	255	0.0610	—	255	0.0020	—	255

TABLE 12-5: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)

BAUD RATE (K)	Fosc = 20 MHz		SPBRG value (decimal)	16 MHz		SPBRG value (decimal)	10 MHz		SPBRG value (decimal)
	KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR	
9600	9.615	+0.16%	129	9.615	+0.16%	103	9.615	+0.16%	64
19200	19.230	+0.16%	64	19.230	+0.16%	51	18.939	-1.36%	32
38400	37.878	-1.36%	32	38.461	+0.16%	25	39.062	+1.7%	15
57600	56.818	-1.36%	21	58.823	+2.12%	16	56.818	-1.36%	10
115200	113.636	-1.36%	10	111.111	-3.55%	8	125	+8.51%	4
250000	250	0	4	250	0	3	NA	—	—
625000	625	0	1	NA	—	—	625	0	0
1250000	1250	0	0	NA	—	—	NA	—	—

BAUD RATE (K)	Fosc = 7.16 MHz		SPBRG value (decimal)	5.068 MHz		SPBRG value (decimal)	4 MHz		SPBRG value (decimal)
	KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR	
9600	9.520	-0.83%	46	9598.485	0.016%	32	9615.385	0.160%	25
19200	19.454	+1.32%	22	18632.35	-2.956%	16	19230.77	0.160%	12
38400	37.286	-2.90%	11	39593.75	3.109%	7	35714.29	-6.994%	6
57600	55.930	-2.90%	7	52791.67	-8.348%	5	62500	8.507%	3
115200	111.860	-2.90%	3	105583.3	-8.348%	2	125000	8.507%	1
250000	NA	—	—	316750	26.700%	0	250000	0.000%	0
625000	NA	—	—	NA	—	—	NA	—	—
1250000	NA	—	—	NA	—	—	NA	—	—

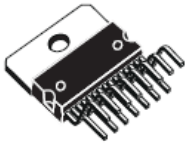
BAUD RATE (K)	Fosc = 3.579 MHz		SPBRG value (decimal)	1 MHz		SPBRG value (decimal)	32.768 kHz		SPBRG value (decimal)
	KBAUD	ERROR		KBAUD	ERROR		KBAUD	ERROR	
9600	9725.543	1.308%	22	8.928	-6.994%	6	NA	NA	NA
19200	18640.63	-2.913%	11	20833.3	8.507%	2	NA	NA	NA
38400	37281.25	-2.913%	5	31250	-18.620%	1	NA	NA	NA
57600	55921.88	-2.913%	3	62500	+8.507	0	NA	NA	NA
115200	111243.8	-2.913%	1	NA	—	—	NA	NA	NA
250000	223687.5	-10.525%	0	NA	—	—	NA	NA	NA
625000	NA	—	—	NA	—	—	NA	NA	NA
1250000	NA	—	—	NA	—	—	NA	NA	NA

Anexo B – Datasheet da ponte H L298.

L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15

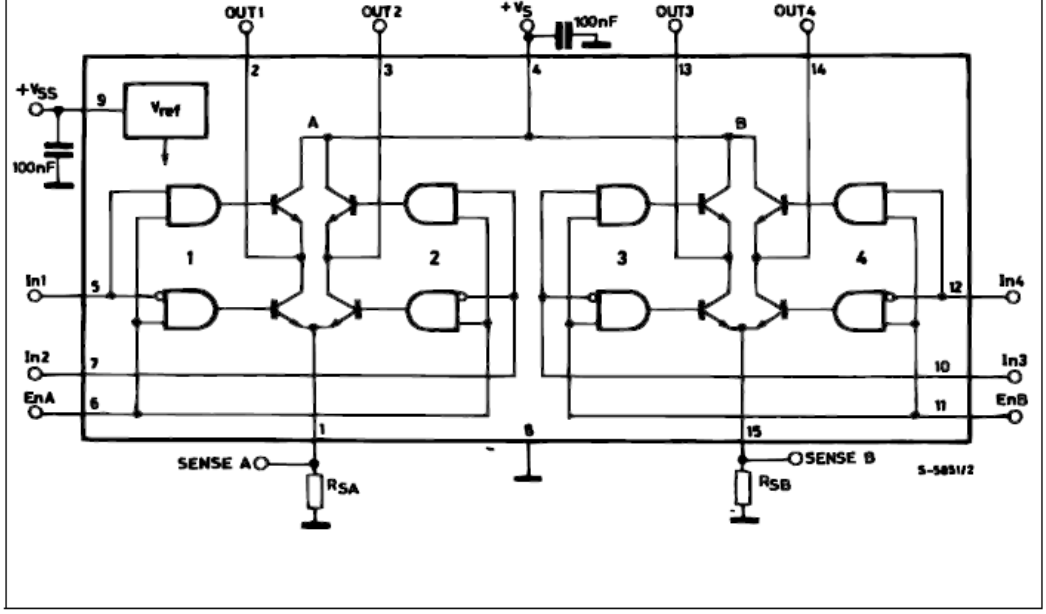
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

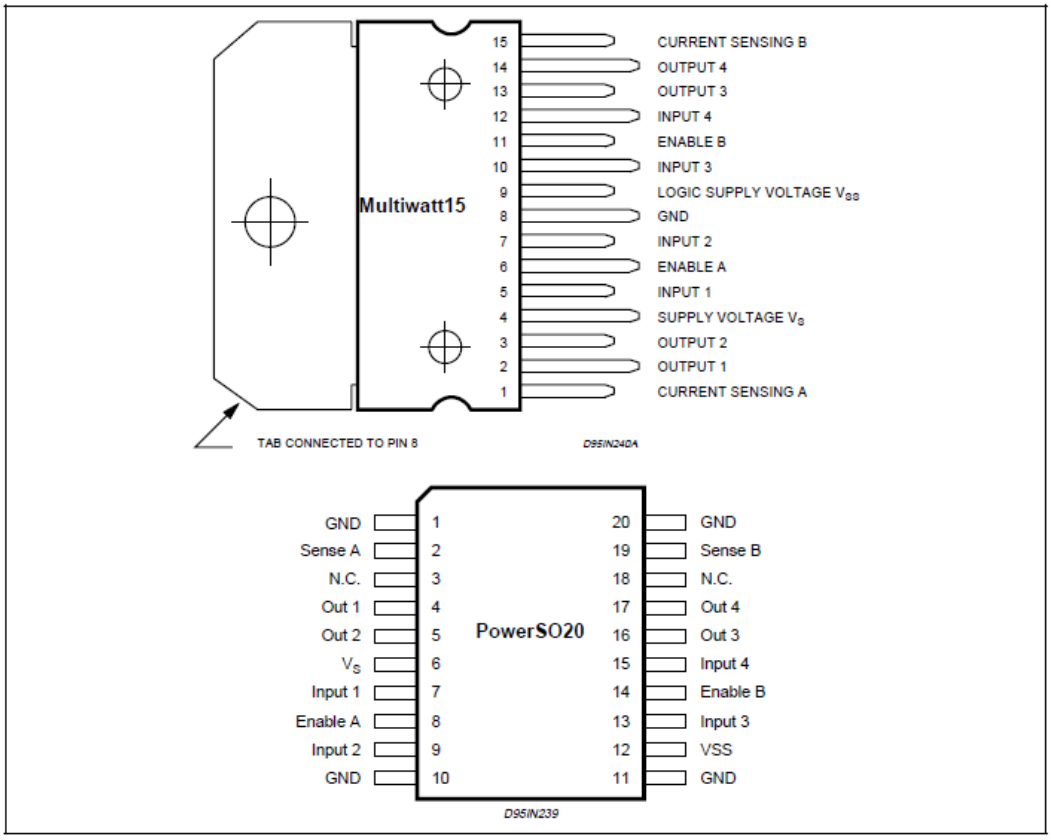
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition	V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0 V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0 V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L V_i = X			6	mA
V_{iL}	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V_{iH}	Input High Voltage (pins 5, 7, 10, 12)		2.3		VSS	V
I_{iL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L			–10	µA
I_{iH}	High Voltage Input Current (pins 5, 7, 10, 12)	Vi = H ≤ V_{SS} –0.6V		30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)		2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L			–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V		30	100	µA
$V_{CEsat\ (H)}$	Source Saturation Voltage	I_L = 1A I_L = 2A	0.95	1.35 2	1.7 2.7	V V
$V_{CEsat\ (L)}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)	0.85	1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)	1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)		–1 (1)		2	V

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V_i)	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
T_2 (V_i)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V_i)	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
T_4 (V_i)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V_i)	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
T_6 (V_i)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V_i)	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
T_8 (V_i)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

1) 1)Sensing voltage can be –1 V for t ≤ 50 µsec; in steady state V_{sens} min ≥ – 0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

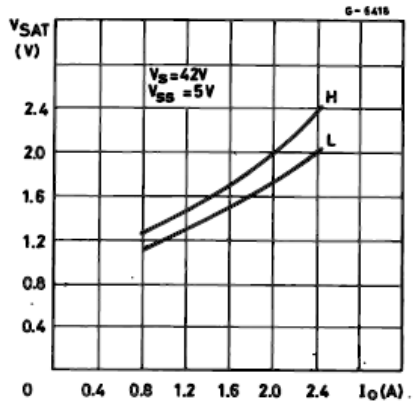
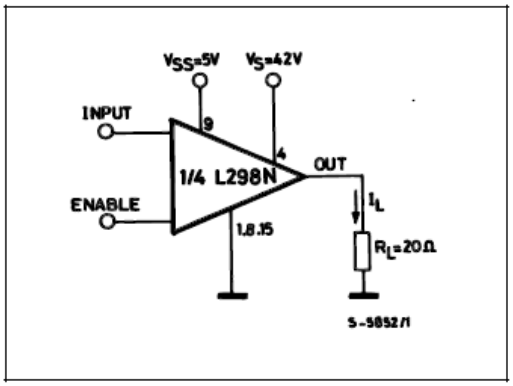


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

Figure 3 : Source Current Delay Times vs. Input or Enable Switching.

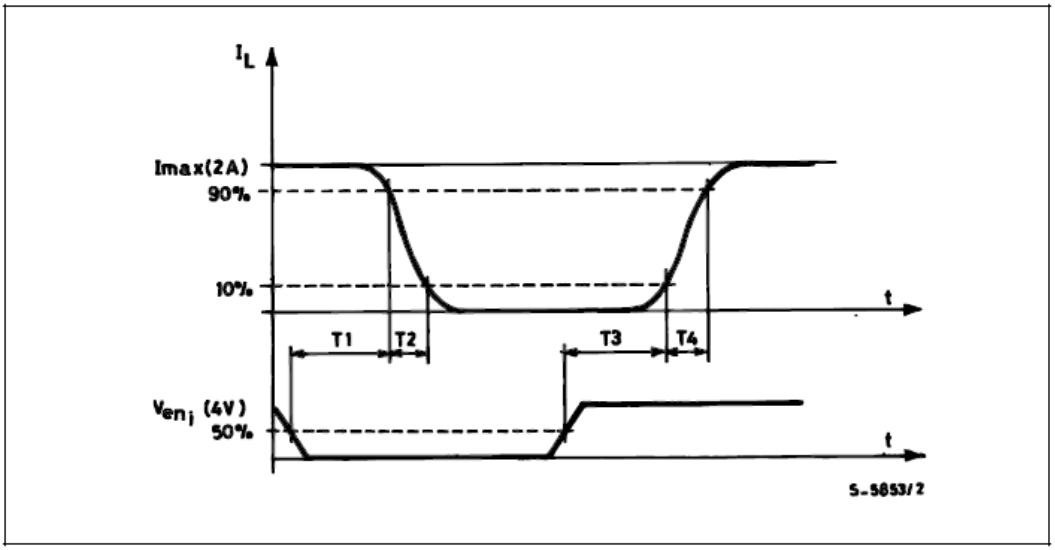
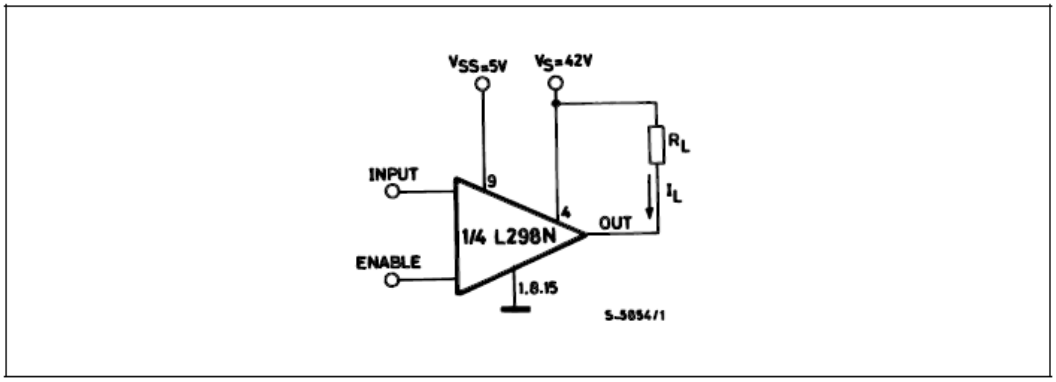


Figure 4 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = L

Figure 5 : Sink Current Delay Times vs. Input 0 V Enable Switching.

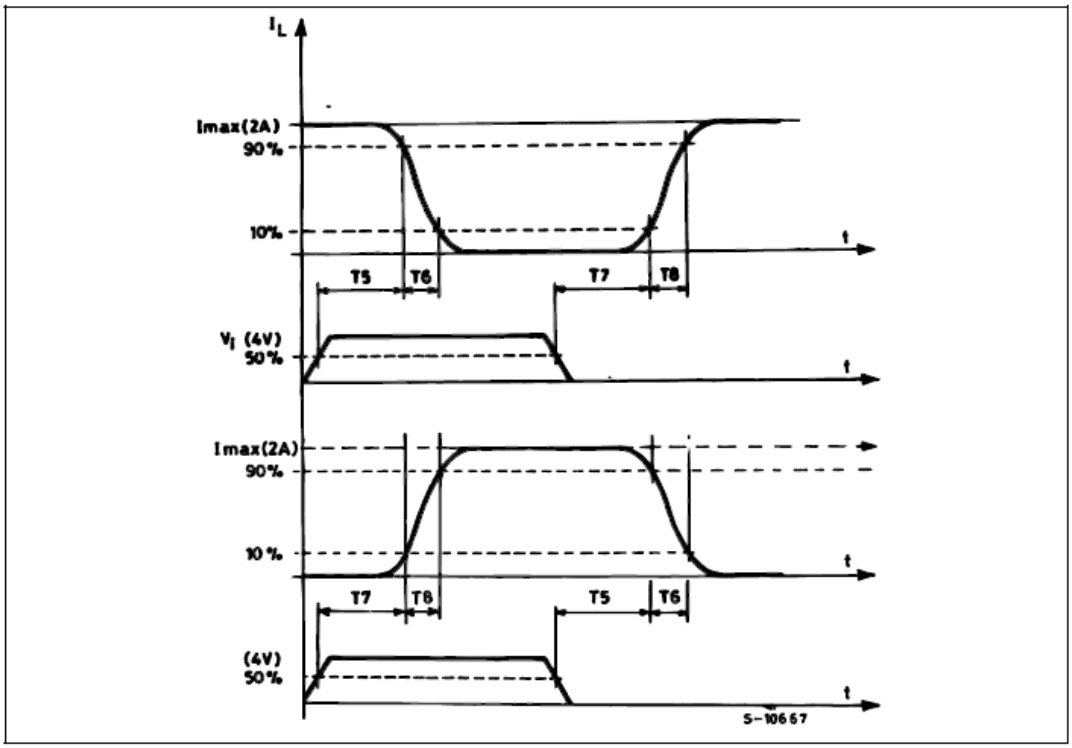
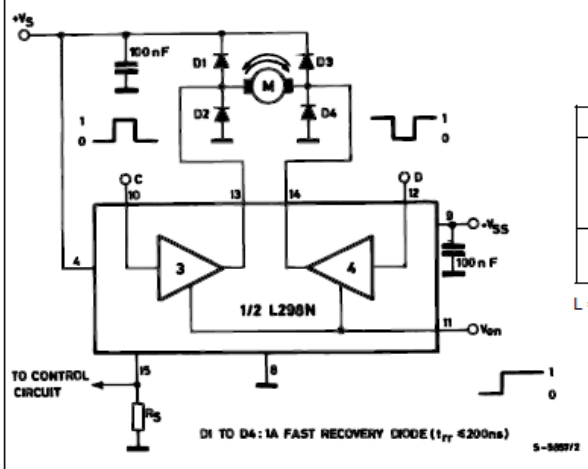


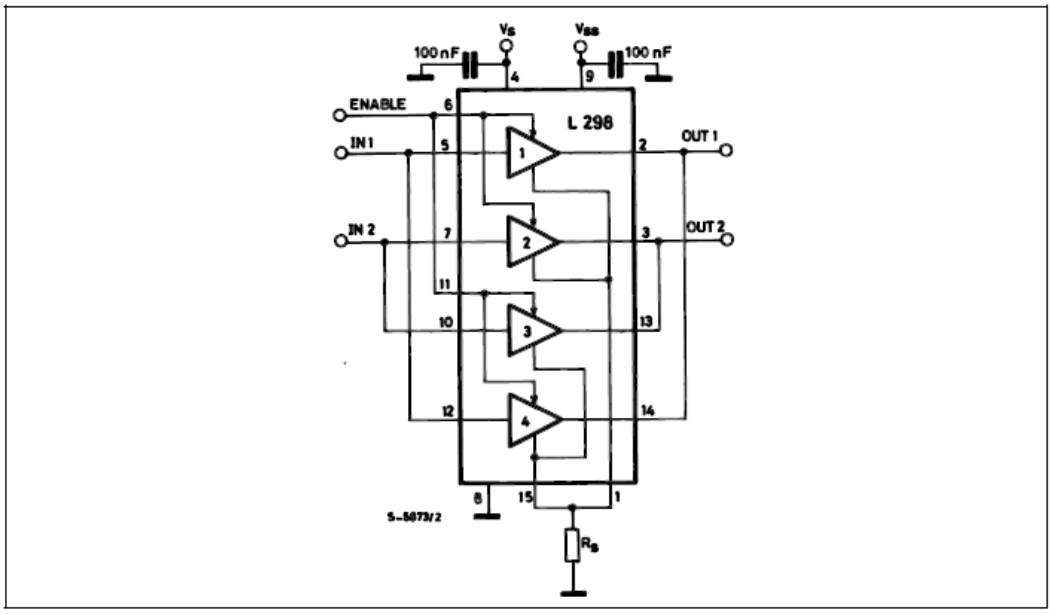
Figure 6 : Bidirectional DC Motor Control.



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

Figure 7 : For higher currents, outputs can be paralleled. Take care to parallel channel 1 with channel 4 and channel 2 with channel 3.



APPLICATION INFORMATION (Refer to the block diagram)

1.1. POWER OUTPUT STAGE

The L298 integrates two power output stages (A ; B). The power output stage is a bridge configuration and its outputs can drive an inductive load in common or differenzial mode, depending on the state of the inputs. The current that flows through the load comes out from the bridge at the sense output : an external resistor (R_{SA} ; R_{SB}.) allows to detect the intensity of this current.

1.2. INPUT STAGE

Each bridge is driven by means of four gates the input of which are In1 ; In2 ; EnA and In3 ; In4 ; EnB. The In inputs set the bridge state when The En input is high ; a low state of the En input inhibits the bridge. All the inputs are TTL compatible.

2. SUGGESTIONS

A non inductive capacitor, usually of 100 nF, must be foreseen between both Vs and Vss, to ground, as near as possible to GND pin. When the large capacitor of the power supply is too far from the IC, a second smaller one must be foreseen near the L298.

The sense resistor, not of a wire wound type, must be grounded near the negative pole of Vs that must be near the GND pin of the I.C.

Each input must be connected to the source of the driving signals by means of a very short path.

Turn-On and Turn-Off : Before to Turn-ON the Supply Voltage and before to Turn it OFF, the Enable input must be driven to the Low state.

3. APPLICATIONS

Fig 6 shows a bidirectional DC motor control Schematic Diagram for which only one bridge is needed. The external bridge of diodes D1 to D4 is made by four fast recovery elements ($t_{rr} \leq 200$ nsec) that must be chosen of a VF as low as possible at the worst case of the load current.

The sense output voltage can be used to control the current amplitude by chopping the inputs, or to provide overcurrent protection by switching low the enable input.

The brake function (Fast motor stop) requires that the Absolute Maximum Rating of 2 Amps must never be overcome.

When the repetitive peak current needed from the load is higher than 2 Amps, a paralleled configuration can be chosen (See Fig.7).

An external bridge of diodes are required when inductive loads are driven and when the inputs of the IC are chopped ; Shottky diodes would be preferred.

This solution can drive until 3 Amps In DC operation and until 3.5 Amps of a repetitive peak current.

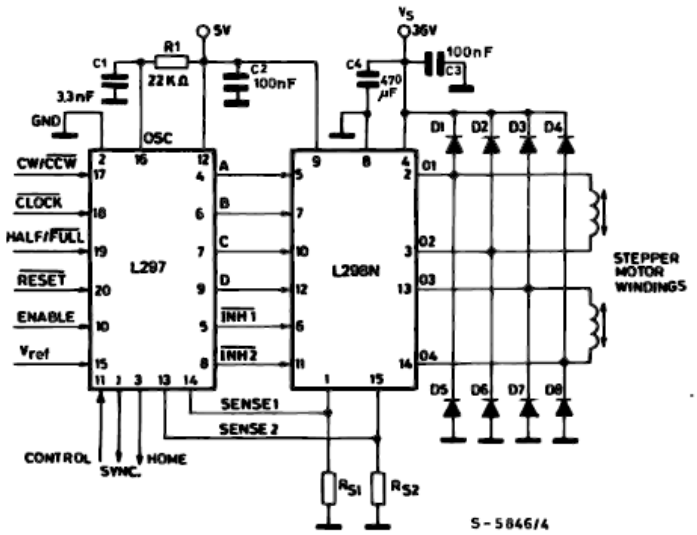
On Fig 8 it is shown the driving of a two phase bipolar stepper motor ; the needed signals to drive the inputs of the L298 are generated, in this example, from the IC L297.

Fig 9 shows an example of P.C.B. designed for the application of Fig 8.

Fig 10 shows a second two phase bipolar stepper motor control circuit where the current is controlled by the I.C. L6506.

Figure 8 : Two Phase Bipolar Stepper Motor Circuit.

This circuit drives bipolar stepper motors with winding currents up to 2 A. The diodes are fast 2 A types.



$R_{S1} = R_{S2} = 0.5\ \Omega$

D1 to D8 = 2 A Fast diodes { $V_F \leq 1.2$ V @ I = 2 A; trr ≤ 200 ns

Figure 9 : Suggested Printed Circuit Board Layout for the Circuit of fig. 8 (1:1 scale).

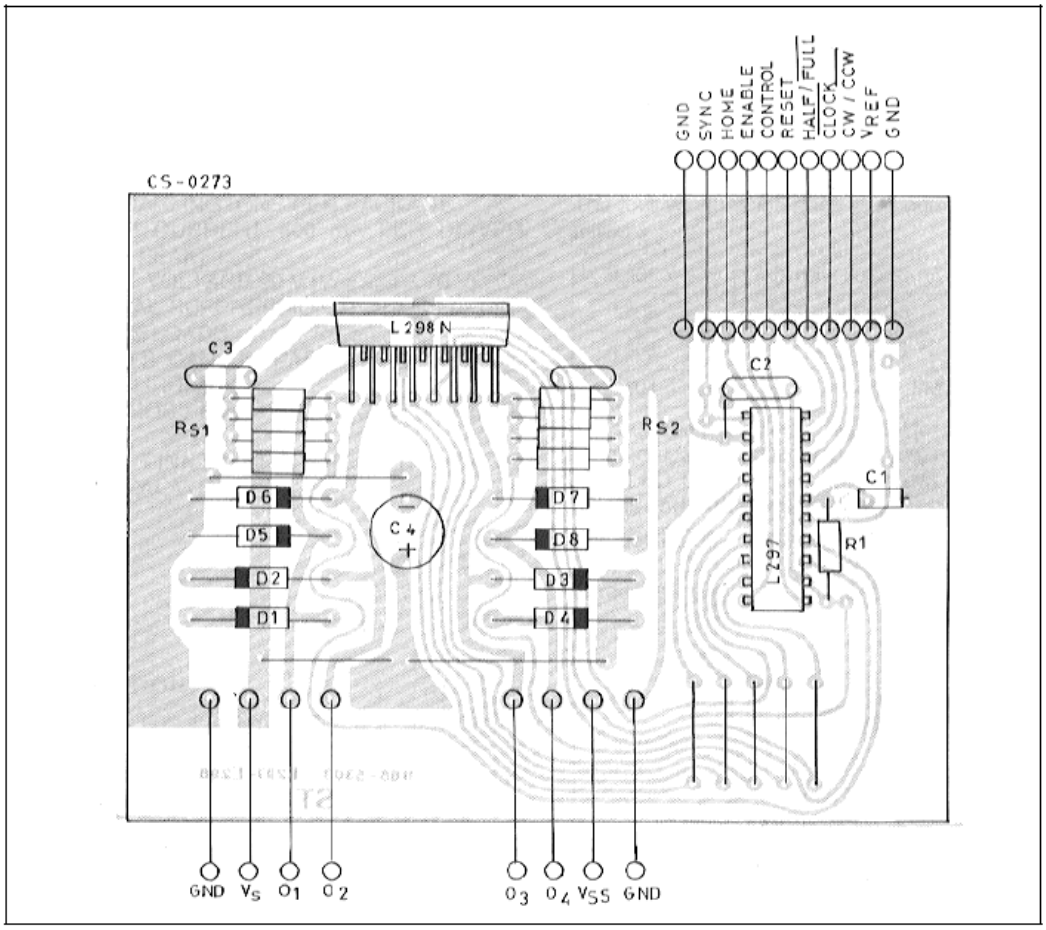
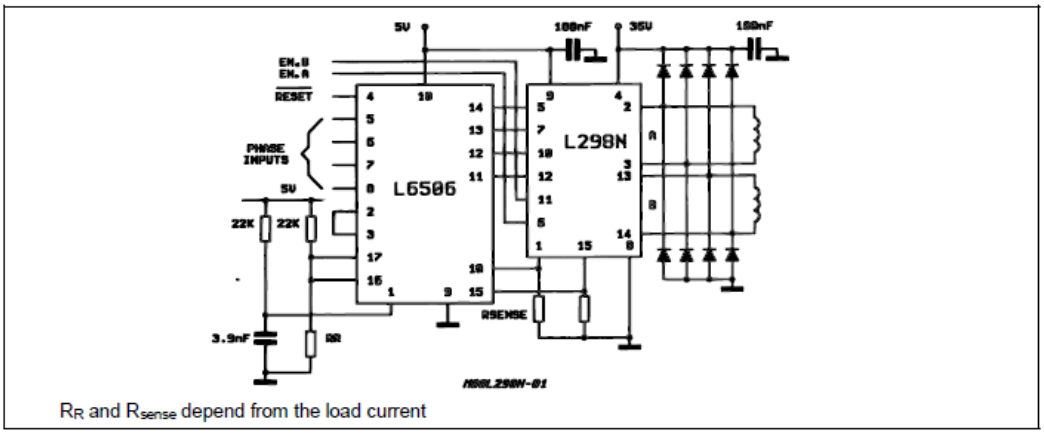


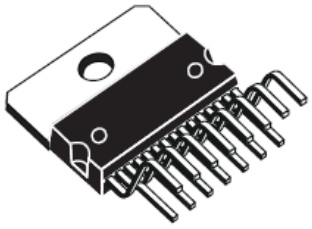
Figure 10 : Two Phase Bipolar Stepper Motor Control Circuit by Using the Current Controller L6506.



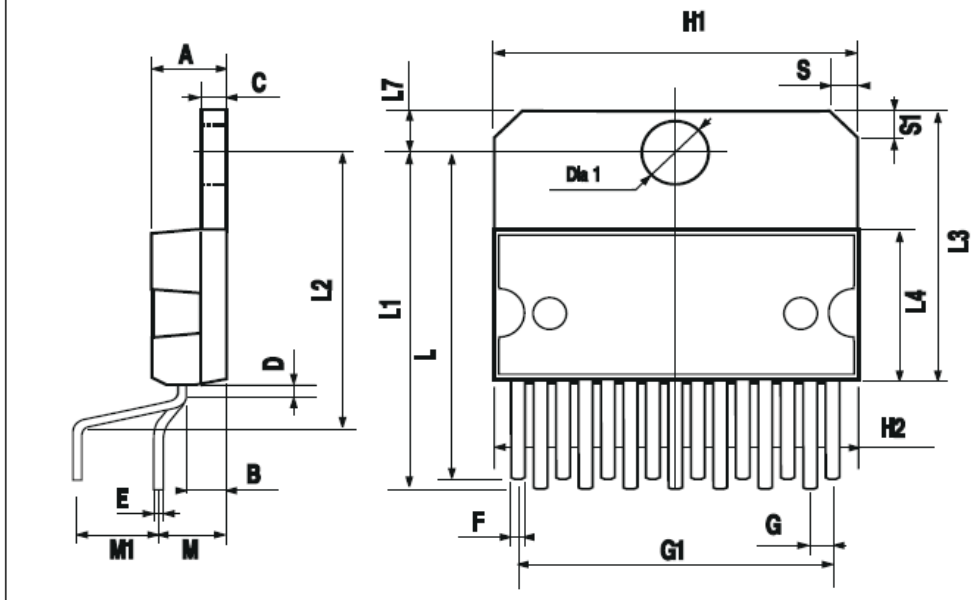
R_R and R_{sense} depend from the load current

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
D		1			0.039	
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.02	1.27	1.52	0.040	0.050	0.060
G1	17.53	17.78	18.03	0.690	0.700	0.710
H1	19.6			0.772		
H2			20.2			0.795
L	21.9	22.2	22.5	0.862	0.874	0.886
L1	21.7	22.1	22.5	0.854	0.870	0.886
L2	17.65		18.1	0.695		0.713
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L7	2.65		2.9	0.104		0.114
M	4.25	4.55	4.85	0.167	0.179	0.191
M1	4.63	5.08	5.53	0.182	0.200	0.218
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA

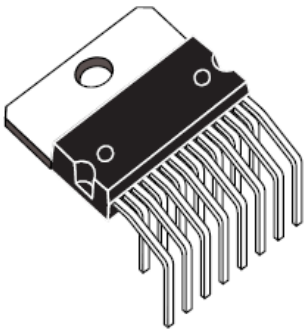


Multiwatt15 V

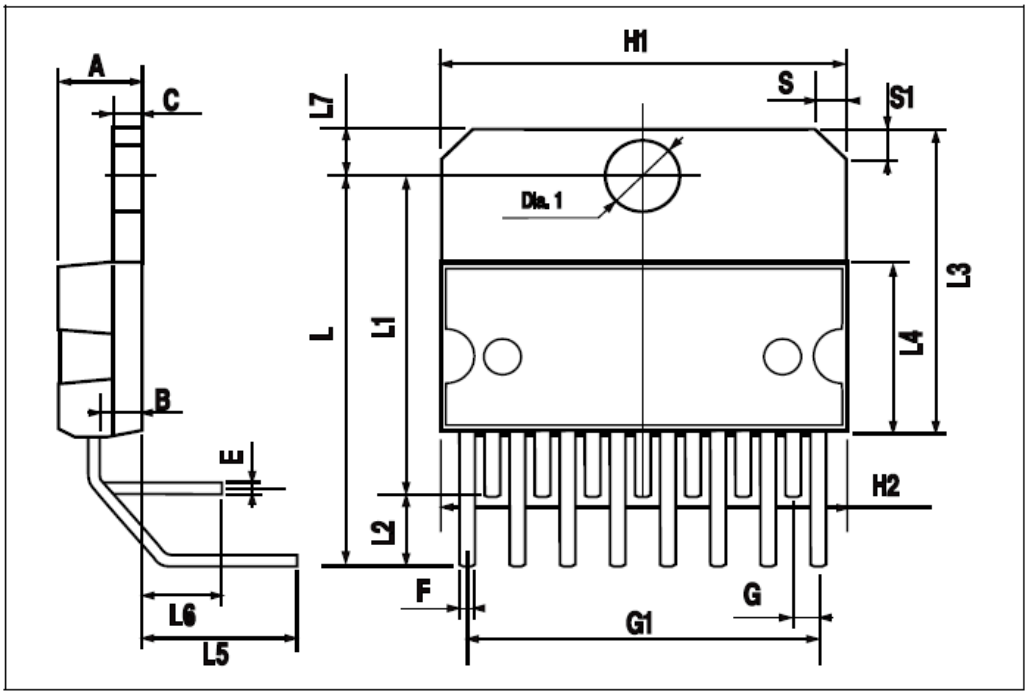


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.14	1.27	1.4	0.045	0.050	0.055
G1	17.57	17.78	17.91	0.692	0.700	0.705
H1	19.6			0.772		
H2			20.2			0.795
L		20.57			0.810	
L1		18.03			0.710	
L2		2.54			0.100	
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L5		5.28			0.208	
L6		2.38			0.094	
L7	2.65		2.9	0.104		0.114
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA



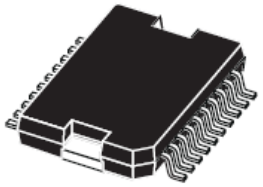
Multiwatt15 H



DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			3.6			0.142
a1	0.1		0.3	0.004		0.012
a2			3.3			0.130
a3	0		0.1	0.000		0.004
b	0.4		0.53	0.016		0.021
c	0.23		0.32	0.009		0.013
D (1)	15.8		16	0.622		0.630
D1	9.4		9.8	0.370		0.386
E	13.9		14.5	0.547		0.570
e		1.27			0.050	
e3		11.43			0.450	
E1 (1)	10.9		11.1	0.429		0.437
E2			2.9			0.114
E3	5.8		6.2	0.228		0.244
G	0		0.1	0.000		0.004
H	15.5		15.9	0.610		0.626
h			1.1			0.043
L	0.8		1.1	0.031		0.043
N	10° (max.)					
S	8° (max.)					
T		10			0.394	

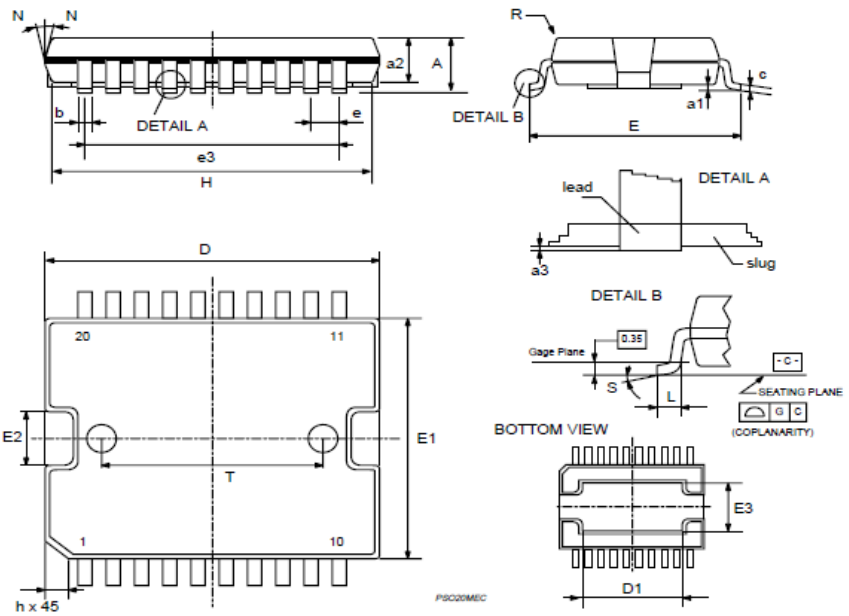
(1) "D and F" do not include mold flash or protrusions.
- Mold flash or protrusions shall not exceed 0.15 mm (0.006").
- Critical dimensions: "E", "G" and "a3"

OUTLINE AND MECHANICAL DATA



JEDEC MO-166

PowerSO20



Anexo C – Programa utilizado no microcontrolador.

```
#include <16f628a.h>                    // identifica microcontrolador alvo
#use delay (clock=20000000)             // define cristal para 20Mhz.
#fuses HS,NOWDT,PUT,NOBROWNOUT,MCLR,NOLVP,NOPROTECT    // bits de configuração (Oscilador, Watchdogtime, temporizador power-up, reset por queda de tensão, Master Clear, programação por baixa tensão, proteção de código)
#fuses HS,NOWDT,PUT,NOBROWNOUT,NOLVP,NOPROTECT         // bits de configuração (Oscilador, Watchdogtime, temporizador power-up, reset por queda de tensão, Master Clear, programação por baixa tensão, proteção de código)
#use rs232(baud = 9600, xmit=pin_b2, rcv=pin_b1)    // habilita a comunicação Serial (velocidade, pino TX, Pino RX)
#define grupo_errado 'G'
#define mestre 'S'
#define seq 'V'
#define comando_errado 'P'
#byte rcsta=0x18            //UART overrun error control variables
#bit Estouro_buffer=rcsta.1     //oerr
#bit ferr=rcsta.2           //ferr
#bit adden=rcsta.3          //adden
#bit cren=rcsta.4           //adjust addressing for processor (cren)
#define led_1 pin_b0
#define led_2 pin_a3
#define led_3 pin_a2
#define led_4 pin_a1
#define led_5 pin_a0
#define Reinicia_buffer cren=0,cren=1


//-----------------------------------------------------------------------------------------------
// programa do robo
//
//-----------------------------------------------------------------------------------------------
char command[5]={'0','0','0','0','0'};                // variáveis auxiliares
int k,k2;
short int g1,g2;

void esvaziarBuffer(){
  char trash;
  while(kbhit())
    trash=getc();
}

void exibe_carac(char N){
  /* led 1 = RB0 (pino 6)
   * led 2 = RA3 (pino 2)
   * led 3 = RA2 (pino 1)
```

```
  * led 4 = RA1 (pino 18)
  * led 5 = RA0 (pino 17)
  */
 switch(N){
  case '0':
output_high(led_1);output_low(led_2);output_low(led_3);output_low(led_4);output_lo
w(led_5);
        break;
  case '1':
output_high(led_1);output_low(led_2);output_low(led_3);output_low(led_4);output_hi
gh(led_5);
        break;
  case '2':
output_high(led_1);output_low(led_2);output_low(led_3);output_high(led_4);output_lo
w(led_5);
        break;
  case '3':
output_high(led_1);output_low(led_2);output_low(led_3);output_high(led_4);output_hi
gh(led_5);
        break;
  case '4':
output_high(led_1);output_low(led_2);output_high(led_3);output_low(led_4);output_lo
w(led_5);
        break;
  case '5':
output_high(led_1);output_low(led_2);output_high(led_3);output_low(led_4);output_hi
gh(led_5);
        break;
  case '6':
output_high(led_1);output_low(led_2);output_high(led_3);output_high(led_4);output_l
ow(led_5);
        break;
  case '7':
output_high(led_1);output_low(led_2);output_high(led_3);output_high(led_4);output_h
igh(led_5);
        break;
  case '8':
output_high(led_1);output_high(led_2);output_low(led_3);output_low(led_4);output_lo
w(led_5);
        break;
  case '9':
output_high(led_1);output_high(led_2);output_low(led_3);output_low(led_4);output_hi
gh(led_5);
        break;
  case 'A':
output_high(led_1);output_high(led_2);output_low(led_3);output_high(led_4);output_l
ow(led_5);
        break;
```

```
   case 'B':
output_high(led_1);output_high(led_2);output_low(led_3);output_high(led_4);output_h
igh(led_5);
         break;
   case 'C':
output_high(led_1);output_high(led_2);output_high(led_3);output_low(led_4);output_l
ow(led_5);
         break;
   case 'D':
output_high(led_1);output_high(led_2);output_high(led_3);output_low(led_4);output_h
igh(led_5);
         break;
   case 'E':
output_high(led_1);output_high(led_2);output_high(led_3);output_high(led_4);output_l
ow(led_5);
         break;
   case 'F':
output_high(led_1);output_high(led_2);output_high(led_3);output_high(led_4);output_
high(led_5);
         break;
   case 'P':
output_low(led_1);output_low(led_2);output_low(led_3);output_low(led_4);output_hig
h(led_5);
         break;
   case 'G':
output_low(led_1);output_low(led_2);output_low(led_3);output_high(led_4);output_lo
w(led_5);
         break;
   case 'Z':
output_low(led_1);output_low(led_2);output_low(led_3);output_low(led_4);output_lo
w(led_5);
         break;
   default :
output_low(led_1);output_high(led_2);output_high(led_3);output_high(led_4);output_h
igh(led_5);
  }
}

void avancar(){
  output_low(pin_b4);output_high(pin_b5);output_low(pin_b6);output_high(pin_b7);
}

void recuar(){
  output_high(pin_b4);output_low(pin_b5);output_high(pin_b6);output_low(pin_b7);
}

void direita(){
  output_high(pin_b4);output_low(pin_b5);output_low(pin_b6);output_high(pin_b7);
}
```

```
void esquerda(){
  output_low(pin_b4);output_high(pin_b5);output_high(pin_b6);output_low(pin_b7);
}

void parar(){
  output_low(pin_b4);output_low(pin_b5);output_low(pin_b6);output_low(pin_b7);
}

void giros(int ct){          // para cada giro
  for (k = 0; k < ct; k++){
    g1 = input(pin_a4);
    if (g1 == 1){
      k--;
    }
    else if (g1 == 0){
      putc('W');
      delay_ms(200);
    }
  }
}

void graus(int ct){          // tempo de giro de 15° = 1/24 de giro
  int ct2;
  ct2 = (ct / 15);
  for (k = 0; k < ct2; k++){
    g1 = input(pin_a4);
    if (g1 == 1){
      k--;
    }
    else if (g1 == 0){
      for (k2 = 0; k2 < 15 ; k2++){
      putc('W');
      delay_ms(3);
      }
      delay_ms(200);
    }
  }
}


void main(){                                          // função principal.
 int i;
 int ct,ctc,ctd,ctu;
 char      cPoss[20]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','P','G','Z','K'};      //
caracteres possíveis.
 char grupo_escolhido='0';                           // variável que armazena o grupo
que controla o veículo.
 output_a(0x00);                                      // desliga todo PORTA.
 output_low(pin_b0);                                        // desliga pinos do led do
PORTB.
```

```
  for(i=0;i<20;i++){                                    // teste inicial do pic.
    exibe_carac(cPoss[i]);                              // exibe os caracteres de teste.
    delay_ms(500);                                      // aguarda 0.5 s para exibir o
próximo caracter de teste.
  }
  exibe_carac(grupo_escolhido);                         // exibe grupo inicial.
  Reinicia_buffer;
  i=0;
  while(true){                                          // laço infinito (obrigatório para
PICs).
    if(kbhit()){                                        // verifica se há comandos no
buffer da porta serial.
      command[i]=getc();                        // atribui caracteres do comando.
      //printf("%c",command[i]);
      i++;                                    // incrementa a posição do buffer de comando.
    }
    else{
      if(Estouro_buffer){                           // se a flag do estouro do buffer estiver
ativa.
        Reinicia_buffer;                        // reinicia o buffer.
        esvaziarBuffer();                       // limpa buffer da porta serial.
        i=0;                                    // reiniciar o buffer de comando.
      }
    }
    if(i>4){                                    // se foram lidos cinco caracteres (um quadro
completo),
      //printf(" ");
       if(command[0] == mestre){                    // se o grupo for o mestre, o Comand é
uma configuração de grupo.
        grupo_escolhido = command[1];               // configura novo grupo.
        exibe_carac(grupo_escolhido);               // exibe novo grupo.
        putc('D');                              // dá a tarefa como realizada
        i=0;                                    // reiniciar o buffer de comando.
      }
      else if (command[0] == seq){                  // se for uma sequência de comandos
        i=0;                                    // reinicia o buffer de comando.
        ctc = ((command[2] - '0') * 100);     // contador: centenas
        ctd = ((command[3] - '0') * 10);      // contador: dezenas
        ctu = ((command[4] - '0'));           // contador: unidades
        ct = ctc + ctd + ctu;                 // contador final
        switch(command[1]){                                     // verifica qual foi o
comando enviado.
          case 'J':  recuar(); giros(ct); parar(); delay_ms(500); putc('D');
                     break;
          case 'K':  direita(); graus(ct); parar(); delay_ms(500); putc('D');
                     break;
          case 'H':  esquerda(); graus(ct); parar(); delay_ms(500); putc('D');
                     break;
          case 'U':  avancar(); giros(ct); parar(); delay_ms(500); putc('D');
                     break;
```

```
        case 'N':  parar(); delay_ms(500); putc('D');
                   break;
        default:  exibe_carac(comando_errado);                  // se o comando enviado não corresponder a nenhum do exitentes exibe comando_errado.
                  delay_ms(500);                                // aguarda 0.5 s.
                  exibe_carac(grupo_escolhido);                 // exibe novamente o grupo_escolhido.
                  command[0]=command[1];                        // desloca o conteudo do buffer de comando.
                  i=0;                                          // reinicia o buffer de comando.
       }
       ct = 0;
     }
     else if (command[0] == grupo_escolhido){                   // se o grupo enviado corresponde ao grupo configurado.
       i=0;                                                     // reinicia o buffer de comando.
       ctc = ((command[2] - '0') * 100);     // contador: centenas
       ctd = ((command[3] - '0') * 10);      // contador: dezenas
       ctu = ((command[4] - '0'));           // contador: unidades
       ct = ctc + ctd + ctu;                 // contador final
       switch(command[1]){                                      // verifica qual foi o comando enviado.
        case 'K':  recuar(); putc('D');
                   break;
        case 'L':  direita(); putc('D');
                   break;
        case 'J':  esquerda(); putc('D');
                   break;
        case 'I':  avancar(); putc('D');
                   break;
        case 'N':  parar(); putc('D');
                   break;
        case 'G':  recuar(); giros(ct); parar(); delay_ms(500); putc('D');
                   break;
        case 'H':  direita(); graus(ct); parar(); delay_ms(500); putc('D');
                   break;
        case 'F':  esquerda(); graus(ct); parar(); delay_ms(500); putc('D');
                   break;
        case 'T':  avancar(); giros(ct); parar(); delay_ms(500); putc('D');
                   break;
        default:  exibe_carac(comando_errado);                  // se o comando enviado não corresponder a nenhum do exitentes exibe comando_errado.
                  delay_ms(500);                                // aguarda 0.5 s.
                  exibe_carac(grupo_escolhido);                 // exibe novamente o grupo_escolhido.
                  command[0]=command[1];                        // desloca o conteudo do buffer de comando.
                  i=0;                                          // reinicia o buffer de comando.
       }
     }
```

```
    else{                                   // se o grupo não corresponder ao
grupo configurado.
      exibe_carac(grupo_errado);            // exibe grupo_errado.
      delay_ms(500);                        // aguarda 0.5 s.
      exibe_carac(grupo_escolhido);                 // exibe novamente o
grupo_escolhido.
      command[0]=command[1];                // desloca o conteudo do buffer de
comando.
      i=0;                          // reinicia o buffer de comando.
     }
   }
  }
}
```