UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PEDRO BAUMGARTNER NETO

PROJETO DE DIPLOMAÇÃO

DRIVER DE MOTOR DE PASSO CONTROLADO POR CLP

Porto Alegre
2011

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DRIVER DE MOTOR DE PASSO CONTROLADO POR CLP

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para Graduação em Engenharia Elétrica.

ORIENTADOR: Msc. Tiaraju Vasconcellos Wagner

Porto Alegre
2011

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PEDRO BAUMGARTNER NETO

DRIVER DE MOTOR DE PASSO CONTROLADO POR CLP

Este projeto foi julgado adequado para fazer jus aos créditos da Disciplina de "Projeto de Diplomação", do Departamento de Engenharia Elétrica e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: Tiaraju Vasconcelos Wagner

Prof. Tiaraju Vasconcellos Wagner, UFRGS

Universidade Federal do Rio Grande do Sul

Banca Examinadora:

Prof. Dr. Alexandre Balbinot, UFRGS

Doutor em Engenharia Mecânica, UFRGS, Porto Alegre, Brasil

Eng. Cláudio Richter, Dexter

Graduado em Engenharia Elétrica, UFRGS , Porto Alegre, Brasil

Porto Alegre, dezembro de 2011.

DEDICATÓRIA

Dedico este trabalho aos profissionais e professores que ajudaram e contribuíram para a minha formação de Engenheiro Eletricista, principalmente ao Engenheiro Cláudio Richter, que serviu para mim como um exemplo de profissionalismo e comprometimento com a qualidade e excelência na prática da Engenharia.

AGRADECIMENTOS

Aos meus pais, que incentivaram e contribuíram com a minha educação.

Aos meus familiares e amigos, pelo companheirismo e lealdade.

Às pessoas que acreditaram em mim e me deram uma oportunidade profissional.

Às empresas Instituto Radiológico Bento Gonçalves, Thyssen Krupp, Actia, AllConverge e Dexter.

À Universidade Federal do Rio Grande do Sul, pela qualidade de ensino.

Aos professores que dedicam sua vida para transmitir, perpetuar e descobrir novos conhecimentos.

À minha namorada Adriana, por todo o seu amor e carinho.

RESUMO

Com esse projeto desenvolveu-se um driver de motor de passo para ser controlado por um CLP da marca Dexter. Partindo que a comunicação deverá ser feita pela porta serial RS-232, utilizou-se um microcontrolador da família 8051 programado em linguagem assembly e a corrente do motor fornecida através de transistores de potência de junção bipolar. O CLP envia um frame de 4 bytes com o movimento do motor codificado e o microcontrolador o executa. Foi desenvolvida a interface de programação a nível de CLP através do uso de macros e diagrama de blocos lógicos. De acordo com os resultados obtidos, esperou-se reunir informações e embasamento para a implementação de um novo produto comercial que poderá fazer parte a linha Dexter de equipamentos de automação.

Palavras-chaves: Engenharia Elétrica. Motor de passo. Controlador lógico programável. CLP. Dexter. Automação Industrial.

ABSTRACT

With this project it was developed a stepper motor drive to be controlled by Dexter branded PLC. Based on that communication should be made by the RS-232 serial port, we used an 8051 family microcontroller programmed in assembly language and motor current supplied by power bipolar junction transistors. The PLC sends a 4 bytes frame with the movement of the motor encrypted. The programming interface was developed at the level of PLC through the use of macros and logical block diagram. According to the results, it was expected to gather information and basis for the implementation of a new commercial product that could be part of the Dexter line of automation equipment.

Keywords: Electrical Engineering. Stepper Motor. PLC. Drive. Dexter. Industrial Automation.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

CI: Circuito Integrado

CLP: Controlador lógico programável

CTS: Clear to send

DSR: Data ser ready

DTR: Data terminal ready

ISP: In system programmer

LSB: Least significant *byte*

MISO: Master input, Slave output

MOSI: Master output, Slave input

MSB: Most significant *byte*

PC: Personal computer

RPM: Revoluções por minuto

RST: Reset

RTS: Ready to send

SCK: Serial clock

SFR: Special Function Register

TCP/IP: Transmission Control Protocol/Internet Protocol

TTL: Transistor-transistor logic

UART: Universal Asynchronous Receiver/Transmitter

1 INTRODUÇÃO

Devido à necessidade de criação de um driver de motor de passo para a linha de equipamentos de automação Dexter, se propôs com esse trabalho desenvolver um estudo para ter como base para a criação de um produto comercial que atenda às características e funcionalidades necessárias para tal produto. Um CLP por si só poderia acionar um motor de passo através de um estágio de potência, porém isso exigiria dedicação do controlador exclusiva para esse fim, o que não condiz com as funções esperadas de um CLP, que é realizar múltiplas tarefas, onde muitas vezes a temporização é um dos fatores críticos. Por isso, a criação de um equipamento que receba ordens do controlador e desempenhe de maneira autônoma toda a geração dos sinais digitais que controlam o movimento do motor deixa a cargo do CLP apenas a programação de que movimento será executado, e fica o driver dedicado à cumpri-los. Além disso, toda a estrutura e ambiente de programação já existente do CLP é usada para codificar o movimento desejado do motor. Essa programação (diagramas de blocos lógicos) é bastante amigável e intuitiva e permite a criação de macros personalizadas para um processo específico.

.

2 O MOTOR DE PASSO

O motor de passo pode ser visto como um motor elétrico sem comutadores. Tipicamente, todos os enrolamentos são parte do estator, e o rotor é ou um ímã permanente, ou, no caso de motores de relutância variável, um bloco dentado de algum material magneticamente permeável. Toda a comutação deve ser feita externamente pelo controlador do motor, e tipicamente, os motores e controladores são projetados para que motor fique fixo em uma posição ou rotacione em um sentido ou outro de maneira precisa com velocidades ou acelerações controladas.

No campo de atuação desse tipo de motor, em algumas aplicações, pode-se escolher entre servo motores e motores de passo. Ambos são eficazes para fazer posicionamento preciso, mas eles diferem em certos aspectos. Servomotores precisam de algum tipo de sistema de controle com retroalimentação negativa analógica. Tipicamente isso envolve um potenciômetro para fornecer o *feedback* da posição do rotor e um circuito de acionamento de corrente para o motor que é inversamente proporcional à diferença entre a posição desejada e a posição momentânea.

Motores de passo, diferentemente dos servo motores, podem ser usados em sistemas de controle em malha aberta, que são geralmente adequados para sistemas que operam com baixas acelerações e cargas estáticas. Em algumas aplicações, o controle de malha fechada pode ser essencial, como para altas acelerações, particularmente se elas envolverem cargas variáveis. Caso um motor de passo em um sistema de controle de malha aberta for sobrecarregado, o conhecimento da posição do rotor pode ser perdido e o sistema poderá ter que ser reinicializado.

A propriedade essencial do motor de passo é de transformar mudanças chaveadas da excitação em incrementos precisos da posição do rotor. Eles são categorizados como máquinas de dupla saliência, o que significa que eles têm dentes de material magneticamente permeável em ambas as partes estacionária e rotatória. Uma seção transversal de uma parte de um motor de passo

é mostrada esquematicamente na Figura 1. O fluxo magnético cruza o entreferro entre as saliências das duas partes do motor. De acordo com o tipo de motor, o fluxo pode ser proveniente de um imã permanente ou de uma bobina por onde passa uma corrente ou uma combinação dos dois. Todavia, o efeito é o mesmo: os dentes sofrem forças iguais e opostas, que tentam alinhá-los e diminuírem o entreferro entre eles. Como mostra o diagrama, a principal componente dessa força, a força normal (n), está tentando fechar o entreferro, mas para os motores elétricos, o componente de força mais útil é a força tangencial (t), que está tentando mover os dentes lateralmente entre si.

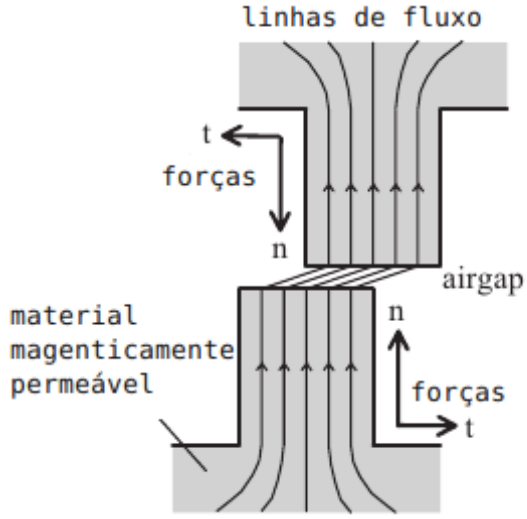


Figura 1 Componentes de força entre dois dentes magneticamente permeáveis.

2.1 FUNCIONAMENTO DO MOTOR DE PASSO

O motor de passo vai converter pulsos elétricos em movimento mecânico de rotação. A rotação do eixo do motor é caracterizada pelo ângulo incremental de passo para cada pulso de excitação. Esse ângulo incremental é repetido precisamente a cada pulso, gerado por um geralmente menor que 5 %, sendo este erro não acumulativo. O resultado é preciso e de movimento fixo, sendo que a cada pulso tem-se o movimento de um único ângulo incremental de passo, o que possibilita um eficiente controle de posição. Se o motor for corretamente dimensionado, obtemos

um motor que não depende da carga, desde que ela imponha um torque sempre menor que o torque do motor.

O circuito excitador é constituído por um circuito sequencial (controlador) e um estágio amplificador de saída. O circuito sequencial pode ser projetado para que o motor gire com diferentes modos de acionamento.

2.2 MODOS DE ACIONAMENTO

De acordo com a sequência de excitação das fases do motor podemos ter alguns modos distintos de acionamento. Será implementado no trabalho a possibilidade de operar nos modo *full step* e *half stet.* Esses modos são descritos abaixo.

2.2.1 PASSO COMPLETO *(FULL STEP)*

Este modo de acionamento se caracteriza pelo fato de que o motor desloca seu rotor em passo completo a cada pulso de acionamento que recebe em suas a fases. O passo completo pode ser ainda realizado no modo normal e no modo *wave*.

2.2.1.1 MODO *WAVE*

Nesse acionamento somente uma fase é energizada por vez a cada passo. Com isso, obtemos o modo com menor consumo, porém o modo com menor torque.

2.2.1.2 MODO NORMAL

Se energizarmos duas fases simultaneamente para cada passo, podemos produzir um torque maior que no modo *wave*. Dois polos adjacentes são magnetizados, o que faz o rotor atingir o equilíbrio em uma posição intermediária entre esses dois polos magnetizados.

2.2.2 MEIO PASSO (*HALF STEP)*

Outro tipo de acionamento possível consiste em energizar, alternadamente, uma e duas fases, permitindo deste modo deslocar o rotor em meio passo de cada vez. Neste modo de acionamento é duplicado o número de passos para completar uma volta e o torque no rotor varia em cada passo, sendo maior quando duas fases estão energizadas. Logo, obtemos uma alternância entre passos fortes e fracos, e o torque disponível é limitado pelo passo mais fraco, com uma melhoria significativa na suavidade do movimento em baixas velocidades.

Como uma alternativa de compensar os torques fracos do modo “half step”, se pode empregar um nível mais alto de corrente quando houver apenas uma fase energizada, igualando ao torque produzido quando há duas fases energizadas.

A tabela 1 resume a sequência de acionamento das fases do motor. No primeira coluna temos a representação da excitação das bobinas por “zeros” e “uns”. O “um” representa a bobina excitada com corrente, e os “zeros” a bobina desenergizada.

Tabela 1 Modos de operação.

Sequência	Nome	Descrição
0001	Modo Wave Uma fase	Consome menos energia. Apenas uma fase é energizada por vez. Assegura precisão posicional, independentemente de qualquer desequilíbrio de enrolamento no motor.
0010		
0100		
1000		
0011	Modo alto torque Duas fases	Alto Torque - Esta seqüência energiza duas fases adjacentes, que oferece um produto torque-velocidade melhorada e maior torque de retenção.
0110		
1100		
1001		
0001	Meio Passo	Duplica a resolução de passos do motor. Mas o torque não é uniforme a cada passo. Esse sequência reduz a ressonância do motor
0011		
0010		
0110		
0100		
1100		
1000		
1001		

2.2.3 MICRO-PASSO *(MICROSTEPPING)*

Verifica-se que se duas fases forem energizadas com correntes iguais produz uma posição de passo intermediária a meio caminho entre as posições em que há uma fase única ligada. Caso as correntes nas duas fases sejam desiguais, a posição do rotor será deslocada em direção ao polo mais forte. Este efeito é empregado no *driver* de micro passo, que subdivide o passo básico do motor estabelecendo uma escala proporcional da corrente nas duas fases. Desta forma, o tamanho do passo é diminuído e a suavidade do movimento em baixas velocidades é sensivelmente melhorada. A forma de onda ideal da corrente nesse tipo de excitação é uma senoide. Cada fase é alimentada com uma onda senoidal deslocada de 90º entre si, eu seja, em quadratura. O passo associado ao movimento normal do motor irá desaparecer nesse tipo de acionamento. Isso ocorre porque as ondas senoidais permitem uma transição contínua de um polo paro o próximo. Enquanto em uma bobina a corrente aumenta, na outra ela diminui. O motor avança suavemente e o torque é contínuo em qualquer posição.

2.3 CARACTERÍSTICAS DOS MOTORES DE PASSO

2.3.1 Conceitos importantes

(a) Fase: Cada uma das bobinas, ou cada uma das metades de uma bobina no caso das que possuem derivação central, que compõem o enrolamento do motor.

(b) Torque de Retenção (*Holding Torque*) : É o torque aplicado ao eixo do motor suficiente para deslocar o seu rotor da posição de equilíbrio (rotor parado e travado pelas forças magnéticas oriundas da interação eletromagnética entre os polos do estator e rotor, quando pelo menos uma das fases do motor está energizada).

(c) Torque Residual (*Detent Torque*) : É o resultado do fluxo magnético permanente que age nos polos do estator, no caso de motores de passo que possuem ímã permanente em seu rotor.

(c) Resposta de Passo : É o tempo de atraso para o motor dar um passo comandado. Esse tempo é função do quociente torque/inércia. Para o motor sem carga é da ordem de milissegundos.

(d) Ressonância: O motor de passo possui uma certa frequência natural característica, sendo que quando o motor atinge esta frequência, ocorre um aumento de ruído e vibração, e o motor pode ainda perder alguns passos ou até oscilar. O valor dessa frequência depende do motor, carga, e circuito driver.

2.3.2 Características Estáticas

As características relacionadas com o motor estacionário são chamadas de estáticas.

(a) Relação T/θ. O motor de passo primeiramente é mantido na posição de equilíbrio por um dos modos de excitação, digamos, fase simples ou dupla. Se um torque externo é aplicado ao eixo, um deslocamento angular ocorrerá. Essa relação entre um torque externo e o deslocamento é mostrada no gráfico da Figura 2. O torque estático máximo que pode ser aplicado ao eixo é chamado de torque de retenção, que ocorre em $\theta = \theta_M$. Deslocamentos maiores que θ_M, o torque estático não atua mais no sentido ,ao ponto de equilíbrio original, e sim para o próximo ponto de equilíbrio. Podemos defini-lo como torque máximo que pode ser aplicado ao rotor de um motor excitado sem causar movimento contínuo.

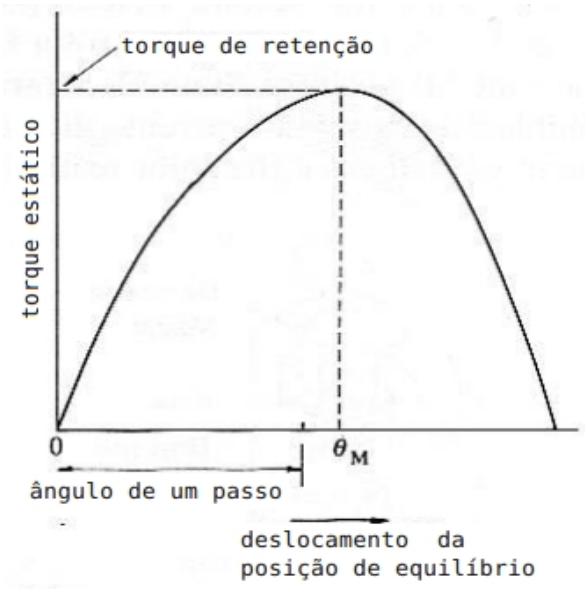


Figura 2 Relação T/θ.

(b) Relação T/I. O torque máximo estático aumenta com a corrente de excitação. A Figura 3 compara uma curva típica de um motor híbrido com a de um motor de relutância variável, com ângulo de passo de ambos sendo de 1,8º.

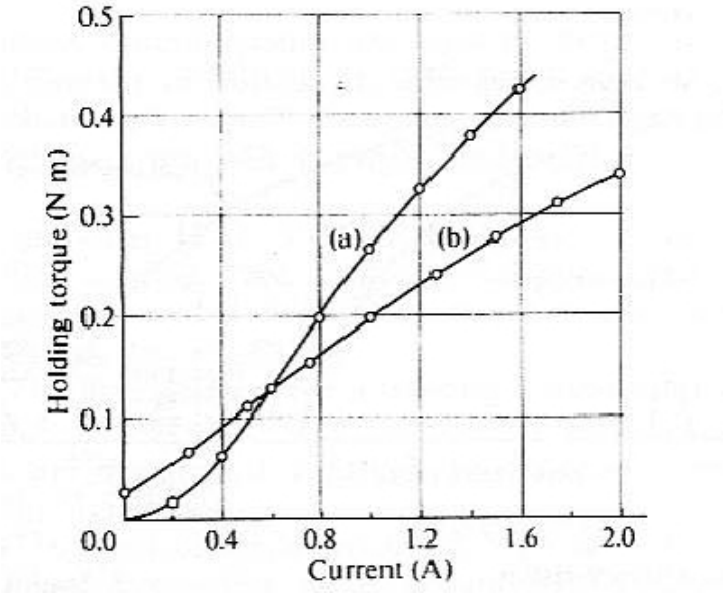


Figura 3 Exemplos de curvas T/I: (a) motor de relutância variável de 4 fases; e (b) Motor híbrido de 4 fases.

2.3.3 Características Dinâmicas

São as características relacionadas ao motor que está em rotação ou está prestes a entrar em movimento.

(a) Características de partida

Referem-se à faixa de torque de carga por atrito em que o motor pode iniciar e parar, sem perder passos, para várias frequências, em um trem de pulsos.

(b) Características de rotação

Depois do motor der a partida, a frequência de pulsos é aumentada gradualmente. O motor irá eventualmente perder o sincronismo. A relação entre o torque da carga e a frequência máxima dos pulsos que o motor consegue sincronizar é chamada de *"pull-out characteristics"*.

(c) Frequência de rotação máxima

É máxima frequência de controle que um motor sem carga pode arrancar e parar sem perder passos.

(d) Taxa máxima de *pull-out*

É definida como a máxima frequência que um motor sem carga pode girar sem perder passos.

(e) Torque máximo de arranque

É o máximo torque que o motor pode acionar e sincronizar com um trem de pulsos em uma frequência baixa de 10Hz.

2.4 Tipos de motores de passo

O esquemático de uma seção transversal de um motor de passo é mostrado na Figura 4

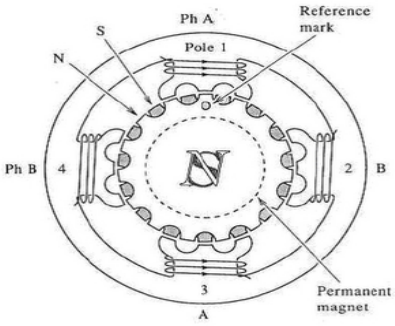


Figura 4 Seção transversal motor de passo.

Os motores de passo existem em uma grande variedade de tamanhos, podemos ter motores acionando pequenos *floppy disks* como também motores que acionam maquinários pesados. São dois os tipo básicos de motores de passo, os motores em que a corrente nunca muda de sentido em suas bobinas, e os motores em que a corrente circula em ambos os sentidos. Esses motores são chamados de unipolares e bipolares, respectivamente.

2.4.1 Motores unipolares

Tipicamente motores de passo unipolares de 5 ou 6 fios tem seus enrolamentos dispostos como no esquemático da Figura 5.

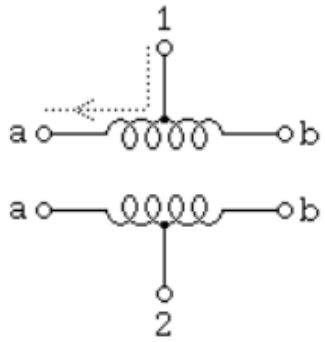


Figura 5 Esquemático dos enrolamentos de motor de passo com 6 fios.

A seta na figura indica o sentido da corrente num dado momento, que é sempre o mesmo em cada segmento da bobina, ou seja, a polaridade magnética na excitação é sempre norte ou sul. Devido a isso, esse tipo excitação é chamado de unipolar, e está representado esquematicamente na Figura 6.

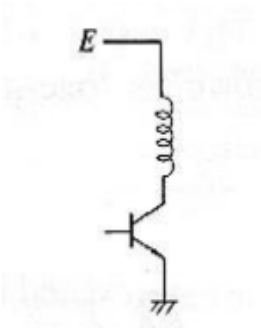


Figura 6 Excitação unipolar.

2.4.2 Motores de passo bipolares

Nesse tipo de motor, a corrente em cada bobina precisa ter seu sentido invertido a fim de ser produzido uma polarização magnética que também troque de sentido. O acionamento das bobinas

exige um hardware mais complexo para tal, sendo que o método mais comum é a ligação em “ponte H”, mostrada na Figura 7.

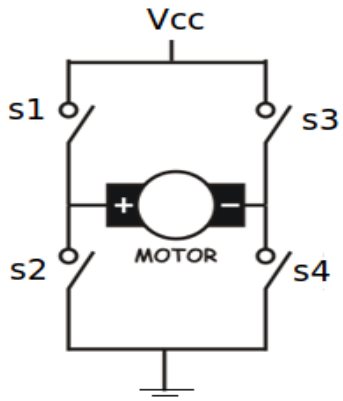


Figura 7 Ponte H.

Esse tipo de motor tem duas ou mais bobinas independentes que circulam corrente nos dos sentido. Na Figura 8 temos a representação esquemática desse tipo de motor.

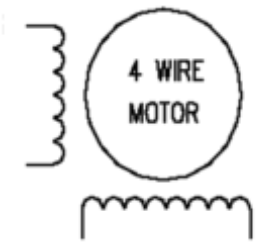


Figura 8 Motor bipolar de duas fases.

2.5 Aplicações dos motores de passo

As aplicações mais comuns dos motores de passo são as que exigem controle preciso de posicionamento e velocidade, principalmente em atividades repetitivas. Na indústria, as aplicações mais comuns são em máquinas rotuladeiras, etiquetadoras, alimentadores de prensa, mesa de

coordenadas, máquinas *router*,, dosadoras, *pick-and-place*, máquinas de laboratório para testes de vida útil, entre outras.

Além da indústria, esse motor é bastante utilizado em periféricos de computadores, como impressoras, *scanners,* entre outros.

2.6 Motor utilizado no projeto

A construção do motor de passo vai definir o seu comportamento em relação a tensão, corrente, torque, velocidade e tempo de resposta. O motor escolhido para o projeto foi um motor unipolar de 6 fios, e tem as seguintes características, segundo o fabricante:

- Tensão do motor: 4.0 V
- Corrente por fase: 2.0A
- Resistência por fase: 2.0 Ohm
- Indutância por fase: 4.4 mH
- Torque: 9,9kgf/cm
- 1.8º por passo

3 Hardware

O diagrama de blocos da Figura 9 resume as principais etapas envolvidas no acionamento do motor.

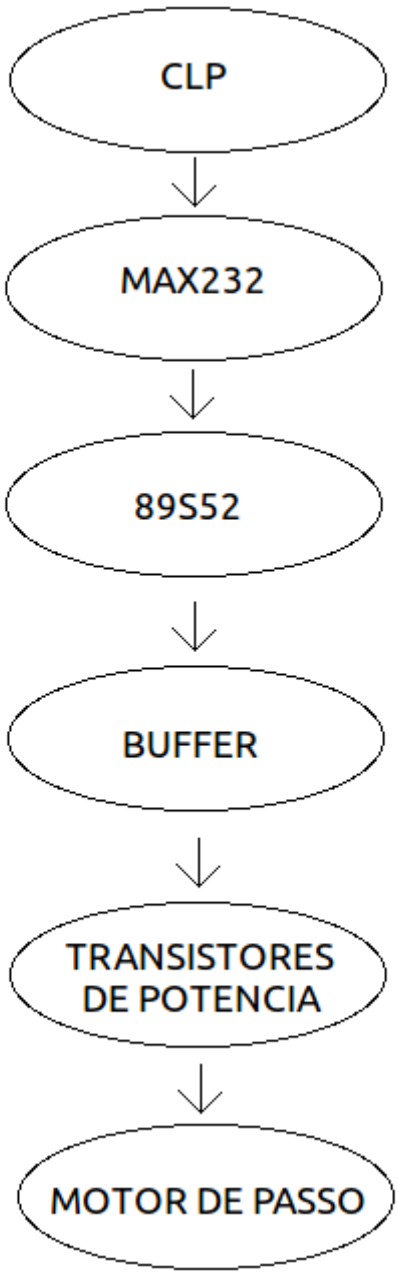


Figura 9 Diagrama de blocos.

Como visto anteriormente, as correntes necessárias para o acionamento das bobinas são da ordem de poucos Ampères e exigem um circuito especial de potência. Com exceção do modo de *microstepping*, basicamente todos os outros modos funcionam chaveando uma fonte de tensão na

bobina de cada fase. Existem várias maneiras possíveis de fazer esse controle “liga e desliga” da corrente, com componentes e topologias de circuitos diferentes. No mercado existem circuitos integrados dedicados a atender o fornecimento de corrente para motores de passo. Eles recebem na entrada o sinal digital proveniente do microcontrolador e na saída fornecem a corrente necessária diretamente nos enrolamentos do estator.

Para esse projeto optou-se por utilizar um circuito comumente usado para essa aplicação, que são transistores funcionando no modo corte e saturação. Em corte, a corrente do transistor entre seus terminais de coletor e emissor é zero e em saturação a corrente que passa depende da corrente de entrada Ib e da tensão Vce. A representação de um transistor NPN com as corrente e tensão, cujo modelo foi usado nesse projeto, está mostrada na Figura 10. O circuito pode ser projetado para que a corrente de saturação seja a corrente nominal de alimentação do motor. Para isso deve-se calcular o valor das resistências de polarização do transistor.

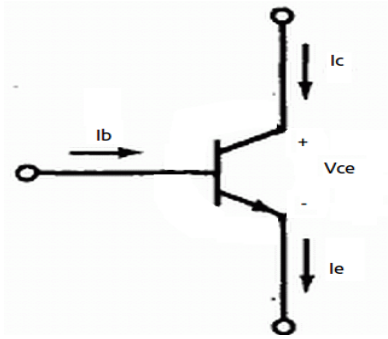


Figura 10 Transistor NPN.

O transistor utilizado no projeto deve suportar uma corrente de 2 A sem sobreaquecimento, pois não serão usados dissipadores de calor. De acordo com as opções possíveis, foi escolhido o TIP 120 que é um transistor de potência de baixo custo e facilmente encontrado no mercado. Esse transistor suporta uma corrente máxima contínua de 5A, logo está bem dimensionado para essa aplicação. Na Figura 10 observa-se a representação desse transistor com a especificação de sua

pinagem. Os terminais indicados como B, C e E representam a base, o coletor e o emissor, respectivamente.

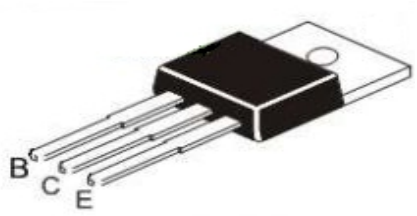


Figura 11 Transistor TIP120.

A topologia de polarização implementada foi a que está esquematizada na Figura 6. Cada fase do motor tem seu circuito de driver de corrente independente dos outros, logo, para o motor em questão, são ao todo 4 circuitos iguais ao da Figura 12.

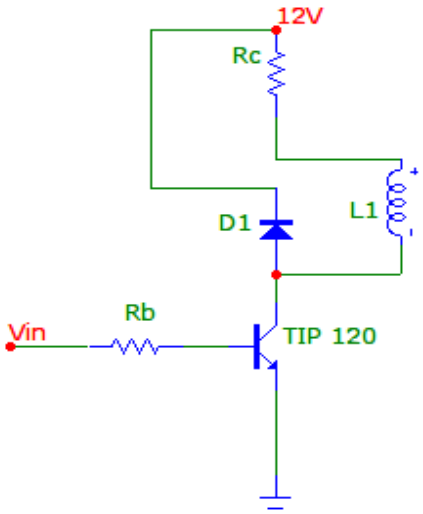


Figura 12 Circuito de polarização do transistor.

A fonte de 12V representada na figura deve ter potência para fornecer a corrente demandada pelo motor. Foi escolhida a fonte de 12V por ser comumente encontrada no mercado. Como a corrente máxima em cada fase do motor é de 2A, a fonte deve ter no mínimo 24W para o modo *wave*. A tensão de entrada Vin é sinal digital gerado pelo microcontrolador, que é um trem de pulsos com 0V ou 5V de amplitude. Quando esse sinal é zero, o transistor está em corte e não há

corrente de coletor. Aplicando os 5V na resistência de base Rb o transistor entra em saturação e conduz. A bobina de cada fase do motor está sendo representada pelo indutor L1.

Como o motor é uma carga indutiva funcionando com transições abruptas de corrente, aplicamos um diodo para desviar a corrente da bobina para a fonte quando o transistor entra em corte.

Essa implementação com uma resistência Rc na prática provoca um aquecimento e um gasto de energia desnecessário. Para a versão comercial desse driver, é util usar os circuitos integrados dedicados que fornecem a corrente demandada pelo motor de maneira eficiente.

3.1 Cálculo das resistência de polarização do transistor

De acordo com o gráfico da Figura 13 retirado do *datasheet* do fabricante do componente TIP120, podemos ver as curvas de corrente de coletor, tendo como parâmetros a corrente de base e a tensão Vce. Como a tensão de operação do motor é de 4V e a alimentação é de 12V, devemos escolher um Vce de saturação que somado com a queda na bobina do motor seja menor que a tensão de alimentação. Como não exite um Vce de saturação de 8V, escolhemos um Vce menor e usamos um resistor no coletor para fornecer a queda de tensão faltante. Precisamos também definir a corrente de base utilizada, que é imposta pela resistência de base. Escolhendo uma corrente de base de 1mA, obtemos, segundo a curva, um Vce de saturação por volta de 1,1V. Os cálculos das resistências são apresentados abaixo.

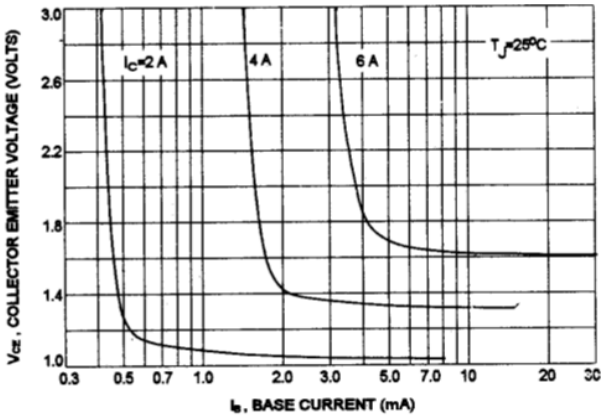


Figura 13 Região de saturação do transistor TIP 120.

O valor da resistência da base será:

$$R_b = \frac{V_i - V_{be}}{I_b}$$

Com $V_i = 5\text{V}, V_{be} = 1{,}4\,V, I_b = 1\text{mA}$, *obtemos* $R_b = 3600\,\Omega$

E escolhemos o valor comercial de $3900\,\Omega$

Para o cálculo do resistor de coletor, partimos do valor da tensão da fonte e do valor do Vce de saturação escolhido.

O valor de R_c será:

$$R_c = \frac{V_{cc} - V_{motor} - Vce}{Ic}$$

$V_{cc} = 12\text{V}, V_{motor} = 4\text{V}$, *e* $V_{ce} = 1{,}1\,V$, *obtemos* $R_c = 3{,}45\,\Omega$

E usamos um valor comercial de resistor de potência de 5Ω .

O valor da corrente de coletor com essa resistência será de:

$$I_c = \frac{V_{cc} - V_{ce}}{R_c + R_{motor}}$$

Como $R_{motor} = 2\Omega, I_c \text{ será de } 1{,}55\ A$.

Utilizamos dois resistores de potência de $10\,\Omega e\, 20\,W$ em paralelo para obter uma resistência equivalente de 5Ω .

Agora, aplicando-se os valores de Rc e Rb utilizados, o cálculo fica:

$$I_b = \frac{V_i - V_{be}}{Rb} I_b = 0{,}923\,mA$$

O gráfico da Figura 13 não apresenta o desenho da curva para uma corrente de coletor de 1,55A, nem podemos determinar com precisão o valor de Vce para essa corrente de coletor e o Ib, logo esse valor foi medido e encontrado uma tensão de 1,18V.

3.2 *Buffer* de tensão

O sinal digital do microcontrolador precisa acionar o transistor de potência TIP120. Para isso, utiliza-se um *buffer* de tensão entre esses dois componentes, para garantir o correto acoplamento de resistências e o fornecimento adequado de corrente para o transistor. Utilizou-se um amplificador operacional na configuração de seguidor de tensão para desempenhar o papel do *buffer,* conforme mostra o esquemático da Figura 14.

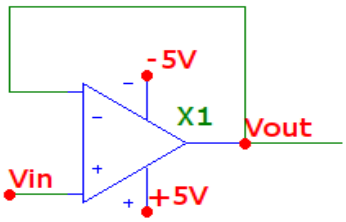


Figura 14 *Buffer*.

O circuito completo precisa de 4 *buffers* de tensão, cada um alimentando a sua fase do motor. Um CI com 4 amplificadores operacionais é necessário para implementar o *buffer* e foi utilizado o modelo TL084, cuja pinagem está esquematizada na Figura 15.

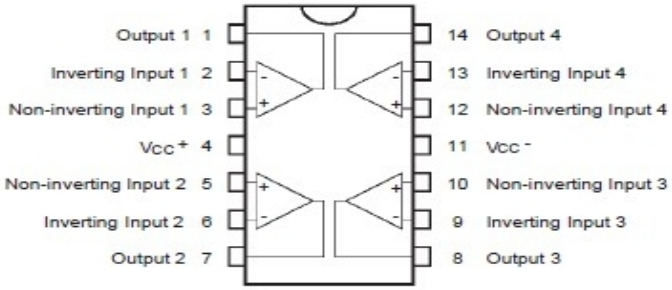


Figura 15 Pinagem do TL084.

Como a amplificação no modo seguidor de tensão tem ganho unitário, é preciso alimentar os amplificadores com +-5V, já que a saída tem valor de 0V ou 5V. Utilizou-se fontes chaveadas para alimentar esse componente.

3.3 Rede RS-232

O controlador lógico programável uDX200 possui uma porta de comunicação serial padrão RS-232. A conexão desse CLP com o PC é feita por essa porta. Embora haja a possibilidade também de programar o CLP usando protocolo TCP/IP, e também via modem, a maneira mais utilizada é ligar direto ao computador através de um cabo com os dois conectores DB9 do tipo fêmea. A ligação do cabo (Figura 16) é do tipo *nullmodem,* que não usa os sinais de controle de fluxo.

Figura 16 Ligação do cabo de comunicação serial.

Então, não são utilizados os sinais de controle da RS-232. Os pinos DTR e DRS, responsáveis pelo controle de fluxo são ligados em curto-circuito, assim como os pinos RTS e CTS, que são utilizado para o *handshake,* também estão ligados em curto-circuito, em ambos os conectores cabo. De acordo com a Figura 17 e a Tabela 2, vemos a disposição e a função de cada pino do conector DB9 macho, que é usado no PC e no CPL Dexter. Notemos que o cabo com dois conectores fêmea conecta o pino 2 do PC no pino 3 do CLP e o pino 3 do PC no pino 2 do CLP.

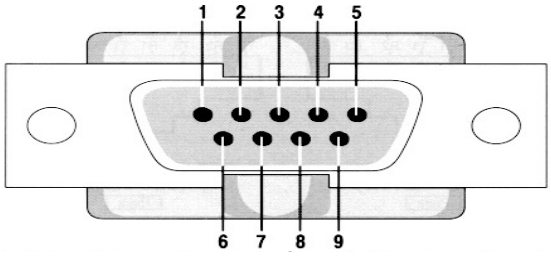


Figura 17 Conector DB9 macho.

Tabela 2 Pinagem conector DB9 macho.

1	Data Carrier Detect
2	RX
3	TX
4	Data Terminal Ready
5	Ground
6	Data Set Ready
7	Request to Send
8	Clear to Send
9	Ring Indicator

De acordo com as funções dos pinos, ligamos em curto-circuito nos conectores fêmeas os pinos 4 e 2 e também os pinos 7 e 8.

3.3.1 MAX232

Para que os níveis elétricos de tensão dos sinais do CLP sejam compatíveis com os do microcontrolador, é necessário usar um CI que faça essa conversão. O padrão RS-232 utiliza tipicamente +/- 15V nos sinais RX e TX, sendo que o nível de tensão negativo representa o nível lógico 1 e o nível de tensão positivo representa o nível lógico 0, numa faixa indicada na tabela 2. Os valores de tensão próximos ao zero não são válidos. De acordo com a Figura 18 vemos o esquemático da ligação do MAX232.

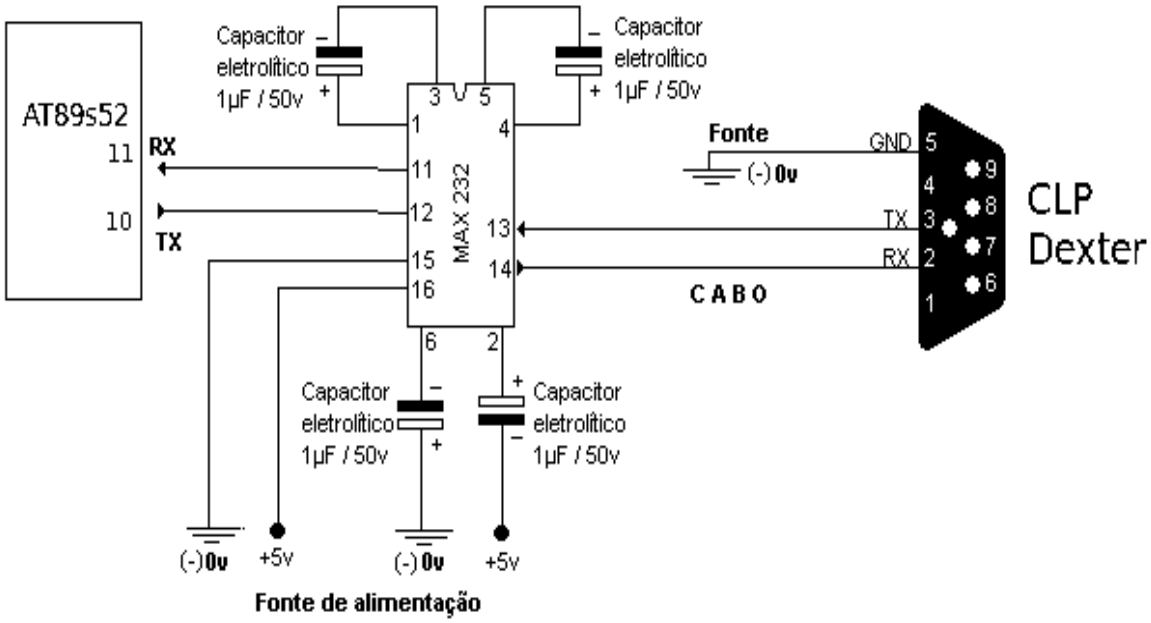


Figura 18 Esquemático da ligação do CI MAX232.

E o esquemático completo do circuito do projeto é visto na Figura 19.

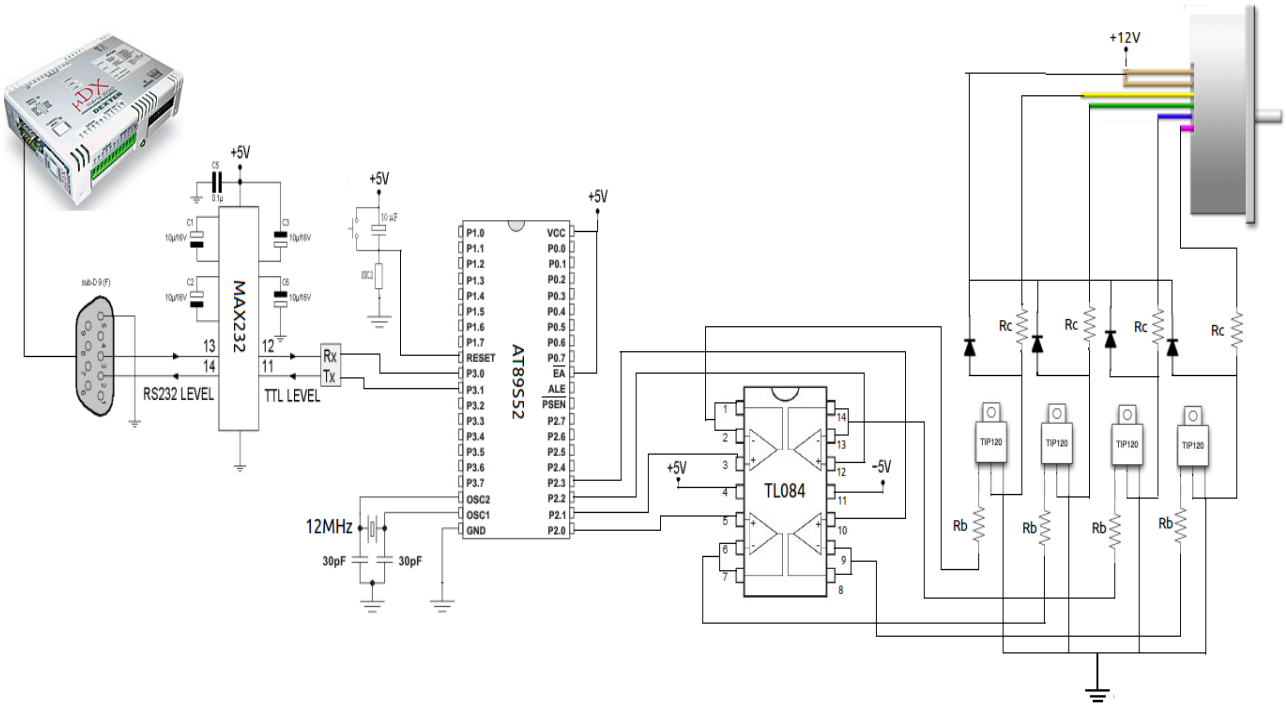


Figura 19 Esquemático completo do projeto.

3.4 Microcontrolador da família do 8051

Mesmo já sendo um microcontrolador antigo, ainda usa-se bastante esse processador em várias aplicações. Com o tempo, foram surgindo opções aprimoradas do 8051 original, sendo adicionado mais memória e alguns recursos. Hoje, o modelo da Atmel AT89S52, usado no projeto, tem 8Kb de memória de programa Flash, funciona com até uma frequência de *clock* de 33Mhz. O preço dele também é uma grande vantagem. A unidade desse *chip* era vendida no varejo por em média R$7,00 em agosto de 2011. Esse valor deve baixar bastante em compras me maior quantidade. A principal limitação desse dispositivo é a antiga arquitetura de 8 bits, o que o torna bem obsoleto quando comparado com a linha PIC da Microchips. Como para essa aplicação não é exigido cálculos que demandem 16 bits, nem acesso a grande quantidade de memória, essa arquitetura continuou sendo uma boa opção.

Outra vantagem é que a gravação do dispositivo é extremamente simples. Um cabo é ligado diretamente da porta paralela do PC para os pinos do microcontrolador. Na Figura 20 temos a representação da pinagem do AT89S52, indicando os pinos de gravação miso, mosi e sck e rst.

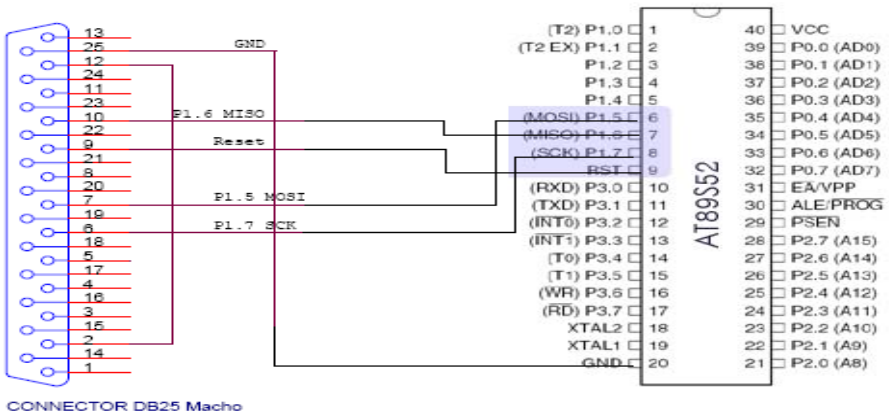


Figura 20 Pinagem AT89S52 e conector DB25.

A alimentação do microcontrolador é de 5V e ele precisa de um cristal externo para o sinal de *clock*. Foi utilizado um cristal de quartzo de frequência de 12 MHz, cuja ligação está esquematizada ma Figura. O pino 31 precisa estar em nível alto para o 8051 executar o programa da memoria interna.

4 SOFTWARE UTILIZADOS

4.1 ISP

Para a gravação do arquivo compilado, foi utilizado o programa ISP 3.0a, cujos pinos de gravação correspondem aos do esquemático da Figura 20. Esse programa é gratuito e está disponível na internet, ele roda em sistemas Windows. Na Figura 21 podemos ver a sua interface.

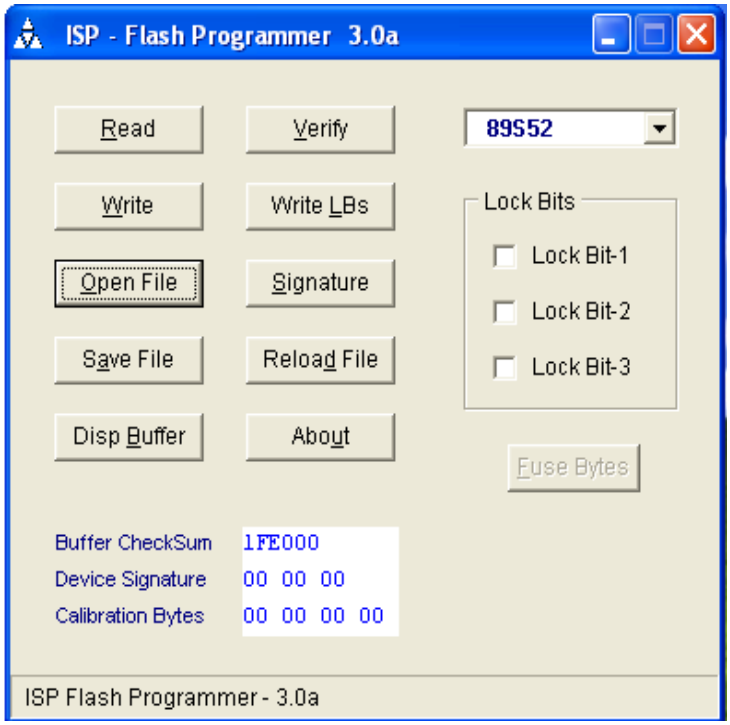


Figura 21 Programa de gravação ISP.

O compilador assembly gera um arquivo binário com formato .hex. Esse arquivo é carregado e transmitido para o microcontrolador que armazena em sua memória de programa interna.

4.2 Software de programação MCU8051

A programação do microcontrolador foi feita toda na linguagem assembly utilizando o programa MCU8051. Esse é um software livre que possui muitos recursos e não há nenhuma limitação quanto ao tamanho do código que pode ser compilado por ele. Ele suporta tanto a linguagem C quanto a linguagem Assembly e tem versões para sistemas Linux, Unix e Windows. A sua interface é mostrada na Figura 22.

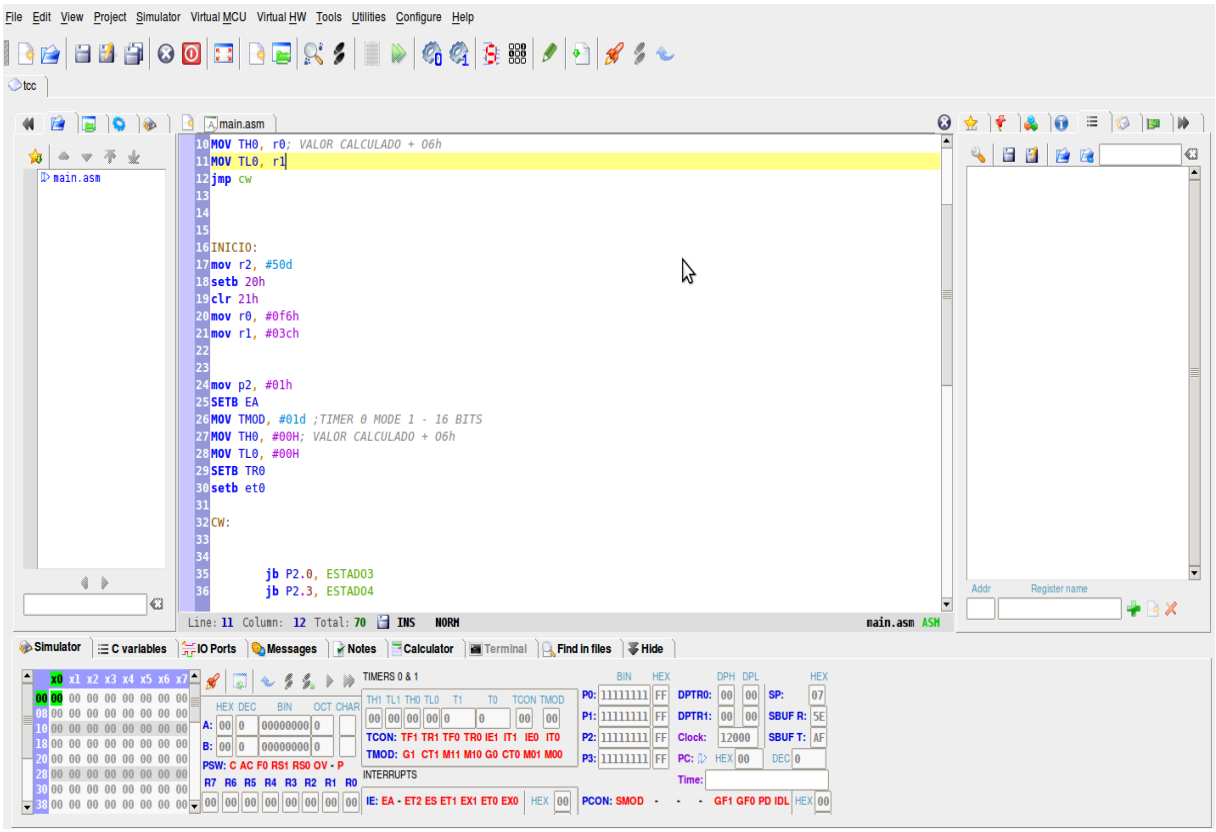


Figura 22 Programa MCU8051.

5 PROGRAMAÇÃO DO AT89S52

A primeira coisa a ser considerada para iniciar-se o código é especificar que recursos do microcontrolador serão usados. Os temporizadores e suas interrupções terão um papel fundamental no programa. Toda a geração de pulsos sequencial é baseada nos *overflows* dos *timers*, assim como a temporização que controla o *baud rate* da porta serial.

Deverá ser definido como se dará a codificação do movimento desejado. Uma sequência de *byte*s constituindo um *frame* será enviado a cada novo comando para o motor. O *baud rate* de comunicação escolhido foi o utilizado por padrão pelo CLP, que é de 38400bps. Além disso, usaremos 1 *stop bit*, 8 bits de dados, sem paridade, sem controle de fluxo e um *start bit.* Ao todo temos 10 bits de transmissão para cada *byte* enviado. Pelo *baud rate* de 38400bps, cada *byte* levará aproximadamente 261us para ter seu envio completo. A interrupção da porta serial só é disparada quando esse *byte* já estiver disponível no *buffer* do microcontrolador.

Basicamente, o sinal enviado pelo CLP irá mandar o microcontrolador gerar a sequência de pulsos desejada, de acordo com o movimento desejado do motor. Na prática, o CLP poderá cooredenar, como um exemplo, um comando para o motor girar, digamos, 5,5 voltas no sentido horário, a uma velocidade angular de 2π rad/s, manter a posição por 3 segundos e girar no sentido anti-horário um número de 5,5 voltas a uma velocidade angular de π rad/s, e repetir o processo a cada 20 segundos. Os *byte*s transmitidos deverão conter a informação codificada dos movimentos. Além disso, deve ser enviado também o modo de operação do motor, se é *full step* ou *half step.* Sendo que o modo *full step* é dividido em modo *wave* e modo normal. Nesse trabalho foi implementado os dois modos de operação em passo completo.

O que define a velocidade de rotação do motor é o tempo em que ocorre a troca de alimentação das fases. O motor utilizado no projeto alcançou uma frequência de revoluções

aproximada de 180 rpm, no modo *full step*, porém nesse limite, o motor fica instável e pode perder sincronia. Diminuindo a velocidade gradativamente, foi observado que por volta de 150 rpm o motor garante movimento sincronizado sem perdas de passos.

Foi limitada a velocidade mínima e máxima do motor de acordo com as limitações impostas pelo hardware utilizado. A velocidade máxima é limitada pela própria inercia do motor e a mínima pelos registradores do *timer*. São dois registradores de 8 bits que juntos formam um *timer* de 16 bits. Esses registradores são incrementados a cada ciclo de máquina, que é o tempo de 12 pulsos de *clock* do 89s52. Como a frequência do relógio é de 12MHz, vamos ter um incremento de uma unidade do *timer* a cada microssegundo. Logo, o maior tempo que podemos contar com um *overflow* do *timer* funcionando nessas condições é de 2^{16}, ou seja 65536 microssegundos. No modo *full step* teríamos uma frequência mínima de revoluções aproximada de 4,57rpm. Para velocidades menores pode-se programar diretamente pelo CLP para enviar a ordem de movimento de um passo por vez.

Visto isso, concluímos que 2 *byte*s na comunicação serão responsáveis pela velocidade de rotação. Tendo a velocidade, precisamos especificar em que sentido se dará a rotação e quantos passos serão. Além disso, existem 2 modos de operação que precisam igualmente serem especificados.

Como os registradores de uso geral do 89s52 tem 8 bits, o número máximo de passos que podemos armazenar em um registrador é de 255. Como temos 200 passos por volta nesse motor, precisamos usar mais de um *byte* para codificar um movimento rotacional maior. Uma solução é usar um *byte* para especificar números inteiros de voltas e outro para especificar números de passos. Assim, por exemplo, para um movimento de 5,25 voltas para um sentido qualquer, um dos *byte*s informa o valor 5 e o outro o valor 50, que corresponde a 0,25 de volta necessário para completar o movimento desejado de 5,25 voltas. Esse *byte* que transmite o valor de 5 voltas pode também codificar o sentido e o modo de operação. Se um dos 8 bits desse *byte* for responsável pelo sentido

de rotação e outros 2 bits codificarem os modos de operação, ainda temos 5 bits para informar o número de votas desejadas. Então podemos ter um *frame de* 4 *bytes* que codifica no máximo 32 voltas em um sentido qualquer, sendo 31 voltas codificadas em um *byte* e mais 200 passos codificados no segundo *byte*, a uma frequência de revoluções por minuto que varia entre aproximadamente 4,7rpm até 150rpm.

Ao todo temos 4 *byte*s que controlam todo o movimento do motor. Esses *byte*s serão enviados em sequência pelo CLP a cada movimento desejado. Em resumo, a Tabela 2 indica as funções que foram atribuídas a cada *byte* do *frame* de comunicação, e o registrador responsável por armazenar o valor recebido.

Tabela 3 *frame* de comunicação.

byte	Função	Registrador responsável
1	Velocidade angular	R0
2	Velocidade angular	R1
3	Sentido, modo de operação e número inteiro de voltas	R2
4	Número de passos	R3

A Tabela 4 resume a função de cada bit que constitui o *byte* 3.

Tabela 4 Descrição do terceiro *byte.*

byte 3	Função
Bit 0	Valor do número de voltas
Bit 1	Valor do número de voltas
Bit 2	Valor do número de voltas
Bit 3	Valor do número de voltas
Bit 4	Valor do número de voltas
Bit 5	Modo de operação
Bit 6	Modo de operação
Bit 7	Bit de modo de posição estática

Irá ser codificados os bits 5 e 6 do *byte* 3 com os seguintes valores estabelecidos na Tabela 5

Tabela 5 Codificação dos modos de operação.

Bit 6	Bit 5	Modo de operação
0	0	Modo *wave* anti-horário
0	1	Modo *wave* horário
1	0	Modo normal anti-horário
1	1	Modo normal horário

O oitavo bit do *byte* 3, chamado de modo de posição estática, tem a função de informar se ao final do movimento rotacional a bobina irá ficar energizada para manter a posição com maior torque, porém com gasto de energia, ou se a bobina ficará desenergizada, logo sem dissipação de potência, e com menor torque de retenção.

5.1 Programação e inicialização de T0

O *timer* 0 tem alguns modos distintos de funcionamento. Foi utilizado o modo 1, que corresponde a um *timer* de 16 bits que incrementa a cada ciclo de máquina. Como foi dito anteriormente, cada ciclo de máquina precisa de 12 pulsos do relógio para se completar. Quando o *timer* 0 estiver com o valor #0FFFF e for incrementado, ele assumirá o novo valor de #0000H, e se as *flags* de interrupções correspondente a esse *timer* estiverem ativadas, o programa irá desviar para a rotina de atendimento de interrupção que, para o *timer* 0 encontra-se no endereço de memória de programa de 000BH. Ali o programa vai desviar para a rotina que gera os pulsos de saída do microcontrolador. Os valores dos registradores TL0 e TH0, que correspondem ao valor inicial do *timer*, são carregados a cada rotina de atendimento da interrupção, de acordo com as valores enviados pela porta serial. Esses valores serão responsáveis pela velocidade angular do motor.

5.2 Inicialização da UART

Para ser possível estabelecer uma comunicação serial, ambos os dispositivos precisam estar programados com o mesmo modo de funcionamento. O padrão RS-232 é uma comunicação serial assíncrona que pode ser configurada para funcionar usando-se apenas 3 fios, sendo esses o RX, TX e *ground*. A velocidade de transmissão é escolhida, assim como o número de bits de dados, o número de *stop* bits, e o uso de paridade. A configuração padrão do CLP Dexter utiliza a taxa de transmissão de 38400bps, 8 bits de dados, 2 *stop* bits e sem paridade. De acordo com esses dados, o microcontrolador AT89s52 foi programado utilizando esses parâmetros, porém com 1 *stop* bit apenas.

A UART do 89s52 tem 4 modos de operação, modo 0 até modo 3. Modos 0 e 2 tem taxas de transferência fixas, o modo 0 é 1/6 da frequência do oscilador e o modo 2 é 1/16 ou 1/32 da frequência do oscilador. Para os modos 1 e 3 a *baud rate* pode ser escolhida e valores típicos são 75, 150, 300, 600, 1200, 4800, 9600, 19200 e 38400bps. Os modos 0 e 1 são usados para conexão entre dois dispositivos. Os modos 2 e 3 são usados para sistemas mestre escravo multiprocessados.

No modo 1 dez bits são usados para especificar um *frame* RS-232 consistindo de um bit de início (lógica 0), 8 bits de dados e 1 bit de parada. A transmissão é feita com o bit menos significativo (LSB) primeiro.

A taxa de transmissão é definida usando um dos temporizadores *onboard* normalmente *timer* 1 em modo 2 ou, quando presente, o *timer* 2. Esse modelo de microprocessador usado no projeto possui o *timer* 2 e este foi usado para gerar a baud rate.

Cálculo da *Baud Rate*:

Como exemplo, determinamos os valores de carregamento do *timer* de acordo com a equação:

$$TH1 = 256 - \frac{2^{SMOD} * Frequênciadooscilador}{192 * baudrate}$$

SMOD =0 para UART no modo 0, 1, 3

SMOD=0 para UART no modo 2, com velocidade 1/32 da frequência do oscilador

SMOD=1 para UART no modo 2, com velocidade 1/16 da frequência do oscilador

Registrador SCON:

O registrador de controle da porta serial possui 8 bits conforme a Tabela 6

Tabela 6 Registrador SCON.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

E os modos de funcionamento são escolhidos de acordo com a tabela 7

Tabela 7 Seleção do modo da comunicação serial.

SM0	SM1	
0	0	Modo 0
0	1	Modo 1
1	0	Modo 2

- SM2 quando igual a 1, permite operação multiprocessada quando nos modos 2 e 3
- REN quando igual a 1, permite o recebimento de dados pela serial
- TB8 usado em operação multiprocessada quando nos modos 2 e 3
- RB8 usado em operação multiprocessada quando nos modos 2 e 3
- TI *flag* de interrupção de transmissão, setado quando o transmissão de um *byte* é completada

- RI flag de interrupção de recebimento, setado quando o *buffer* da serial recebeu o *byte*.

Foi utilizado o modo de funcionamento 1 da UART, que corresponde a 8 bits de dados e com taxa de transferência baseada na frequência de *overflows* do *timer* 2. Para chegarmos ao valor exato da taxa de transferência de 38400bps, é preciso configurar o *timer* 2 como *baud rate generator,* que utiliza valores recarregamento automático armazenados nos SFR RCAP2H e RCAP2L.

A equação para chegarmos ao valor de recarregamento, de acordo com o *baud rate* desejada é a seguinte

$$RCAP2 = 65536 - \frac{frequênciadooscilador}{32 * baudrate}$$

Onde chegamos a um valor de 65526,234 para RCAP2. Como precisamos sempre de valores inteiros, o valor arredondado 65526 proporciona uma *baud rate* de valor real de 37500 e um erro de -2,34%, mas que é um valor tolerável e não inviabiliza a comunicação.

Timer 2:

A operação do *timer* 2 é controlada quase que completamente pelo SFR T2CON, no endereço C8H. Esse SFR é endereçavel bit a bit. Na tabela 7 temos o resumo da função de cada bit.

Tabela 8 Registrador T2CON.

TCLK	Quando este bit for setado, Timer 2 será usado para determinar taxa de recepçãoo da porta serial. Quando zero, Timer 1 será utilizado.
EXEN2	Timer 2 Ativar externo. Quando definido, uma transição 1-0 sobre T2EX (P1.1) fará uma captura ou recarregar a ocorrer.
TR2	Quando setado, o timer 2 dispara. Quando 0, não conta.
C/T2	Se 0, timer 2 é um contador de intervalo, se 1, timer 2 é incrementado por transição 1-0 na P1.0
CP/RL2	Se 0, recarrego automático ocorre no overflow do timer 2 ou na transição 1-0 do T2EX

Resumindo, a inicialização e configuração da porta serial será feita tendo em vista a velocidade de comunicação de 38400bps, 8 bits de dados e 1 *stop* bits. O gerador da *baud rate*, então será o *timer* 2, no modo a*uto reload*, que funciona usando os valores armazenados nos registradores RCAP2L e RCAP2H, sempre que o *timer* 2 atinge o *overflow*. Ao escrever o valor binário 00110100 no SFR T2CON, o *timer* já estará configurado para a função. Devemos ainda mover o valor 0FFh para o registrador RCAP2H e o valor 0F6h para o registrador RCAP2L.

5.3 Geração do sinal sequencial

São 4 pinos que geram os 4 pulsos de saída. Para o programa saber em qual fase do motor será o próximo pulso, será testado o valor de uma variável booleana e mais o estado atual da saída da P2. As variáveis booleanas localizam-se no conjunto de registradores que permitem a leitura e escrita bit a bit na memória RAM do microprocessador. Definimos os bits 20h, 21h, 22h, 23h para armazenar a informação do modo de operação e o sentido de rotação. Só existe uma sequência possível de pulsos que permite um movimento correto. Essa sequência, aplicada de trás pra frente faz o movimento no sentido contrário. Da porta P2 saem os 4 sinais dos pulsos, provenientes dos pinos P2.0, P2.1, P2.2 e P2.3. Logo, no modo *wave*, temos os valores de escrita na porta P2, no sentido anti-horário, como sendo de 1, 2, 4 e 8, correspondendo ao valores binários no *nibble* menos significativos como sendo 0001, 0010, 0100, e 1000, respectivamente. Com isso criamos os 4 estados do modo *wave*. Digamos que, após um certo movimento, o estado final na P2 esteja com o valor 0100. Só há dois estados possível a seguir. Se o motor estiver girando no sentido horário o próximo valor será 1000.

Foi definido por conveniência o nome dos estados de acordo com as cores dos fios que alimentam o motor de passo. As cores são vermelho, laranja, amarelo e azul. A sequência no sentido anti-horário fica, considerando que está sendo partido de uma posição de *reset* conhecida, como sendo de VERMELHO → LARANJA → AMARELO → AZUL.

Se por acaso o movimento desejado for no sentido contrário, a sequência seria VERMELHO → AZUL → AMARELO → LARANJA, repetindo a mesmo sequência novamente para um movimento contínuo.

De acordo com os modos de operação, que podem acionar mais de uma fase ao mesmo tempo, criou-se outros estados agrupando-se cada estado com o seu estado adjacente. Ainda temos 4 estados só que agora somamos os valores de dois estados vizinhos e criamos um novo estado, de acordo com a Tabela 9.

Tabela 9 Estados das saídas.

ESTADO	VALOR
VERMELHO	1
LARANJA	2
AMARELO	4
AZUL	8
VERMELHO + AZUL	9
VERMELHO + LARANJA	3
LARANJA + AMARELO	6

A segunda coluna da tabela 8 mostra o valor que será escrito na porta P2 para acionar a corrente nos fios cujas cores são especificadas na primeira coluna.

Usamos o mesmo algoritmo usado no modo *wave* para o modo normal, a diferença é somente os valores de escrita de cada estado que mudam.

5.3.1 Funcionamento do programa no microcontrolador

O programa funciona basicamente em um *loop* que espera continuamente o sinal de duas fontes de interrupção. A interrupção do *timer0* inicia a sequência de instruções que procura a maneira com que se dará o passo seguinte. Primeiramente é feita a leitura dos registradores que armazenam a quantidade de passos ou de voltas a serem cumpridos. Depois é feita uma leitura nos valores dos bits que armazenam o modo de operação a ser realizado o movimento. Feito isso, o

passo é realizado, tendo em vista o estado atual da saída e o sentido a ser girado, que define o próximo valor de escrita na P2.

A outra interrupção disparada é a da comunicação serial. Quando um *byte* é recebido e o *buffer* de recepção estiver pronto, inicia-se a sequência de leitura dos *byte*s provenientes do CLP.

5.3.2 Interrupção da porta serial

Ao final do primeiro *byte* enviado pelo CLP, temos o disparo da interrupção. O valor recebido é passado ao registrador R0, que armazena o valor que irá para o TH0. Depois, o programa irá aguardar o recebimento do segundo, terceiro e quarto *byte*. Terminado o recebimento dos *byte*s, deve se iniciar o movimento do motor por eles codificados. O valores dos dois primeiros *byte*s e do último *byte* são passados diretamente para para os registradores . O terceiro *byte* carrega dois tipos de informações diferentes e precisa ser feita uma tratativa nesse dado. Primeiramente, se irá separar em dois registradores o valor do terceiro *byte*. Como o número de voltas completas usa os 5 bits menos significativos do *byte* 3, esse valor é registrado em um registrador eliminando-se os 3 bits mais significativos. Se for feita a operação de "e" lógico entre o valor do *byte* e o valor 31 (00011111 em binário), qualquer "1" existente nos 3 bits mais significativos serão perdidos e o *byte* passará a carregar a informação do número de voltas completas apenas. O valor original do *byte* 3 fica armazenado em outro registrador onde irá ser aplicadas operações de rotação do *byte* à esquerda com *carry*. A função é RLC (*Rotate left with carry*) só pode ser aplicada ao acumulador, pois esse gera *carry* à esquerda quando rotacionado por essa operação. À medida que aplicamos essa função, o resultado do *carry* é testado e o programa é desviado para a sub-rotina necessária. Se na primeira rotação já for detectado um *carry*, sabemos que ao término do movimento do motor, a bobina deverá manter-se energizada. Se não houver *carry*, o motor será desernegizado no final da rotação. Os bits de *carry* seguintes irão decodificar os modos de operação setando as variáveis booleanas que serão testadas na interrupção do *timer*0.

4.3.3 Interrupção do *timer*0

A cada *overflow* do *timer* 0 é feita a leitura da velocidade com que se dará o próximo passo. Isso é feito diretamente passando os valores enviados pelo CLP para os registradores TH0 e TL0. Depois, é feita a leitura dos registradores que armazenam do número de passos que o motor deverá cumprir. Caso exista algum valor diferente de zero nesses registradores, eles serão decrementados de uma unidade e o motor irá executar um passo. Para saber qual o modo de operação e qual sentido, será feita uma leitura nos bits que armazenam essa informação.

O primeiro teste realizado é no registrador R4, que diz quantos passos deverão ser cumpridos. Caso haja um valor diferente de zero, ele será decrementado e um passo irá ser feito. Se o valor dele for zero, irá ser feita a leitura do registrador que armazena a informação do número completo de voltas. Se esse valor for diferente de zero, irá ser movido o valor de 200 para o registrador R4, que corresponde a uma volta completa.

A etapa seguinte é a leitura dos bits que armazenam o sentido e o modo de operação que desviarão para as sub-rotinas responsáveis pela escrita da porta P2.

5.4.3 Escrita da P2

São quatro sub-rotinas que escrevem na porta P2. Elas estão resumidas na Tabela 10

Tabela 10 Sub-rotinas de escrita da porta P2.

wcw	modo wave clockwise (sentido horário)
wccw	modo wave counterclockwise (sentido anti-horário)
ncw	modo normal clockwise
nccw	modo normal counterclockwise

Quando o programa é desviado para essas sub-rotinas, são testados os bits um a um da porta P2. Quando o primeiro com o valor 1 for detectado, significa que o próximo estado da porta P2 pode ser determinado, de acordo com o sentido de rotação e o modo de operação.

Todo o código do programa está incluído no anexo e acompanha comentários descrevendo o funcionamento principal.

5.4 Programação do CLP

Para a programação do CLP, em cada movimento desejado, só é necessário enviarmos 4 *byte*s em sequência, codificados da maneira já estipulada.

Utilizou-se o recurso de macros para criar uma estrutura em que só se precise entrar com os valores descritivos do movimento desejado e o programa executa a transmissão dos *byte*s. É preciso observar o tipo de variável e o formato de entrada dos valores para um correto movimento, pois essa macro não prevê erro de entrada de dado. Na Figura 22 está esquematizada a programação utilizada para essa macro.

A macro possui apenas uma saída que gera uma borda de subida ao final da transmissão dos 4 *byte*s. Isso pode ser útil, principalmente em tarefas onde o tempo de execução é um dos fatores críticos.

Na Figura 23 mostra um exemplo de utilização de dessa macro. Foi adicionado comentários no código para explicar ao usuário a função de cada entrada. Para um movimento mais complexo, deve-se encadear várias macros como a da Figura e observar o tempo de execução do movimento do motor para enviar um novo comando. Se um comando for enviado e o motor estiver executando um movimento, o novo comando sobrescreverá o que o motor estiver executando.

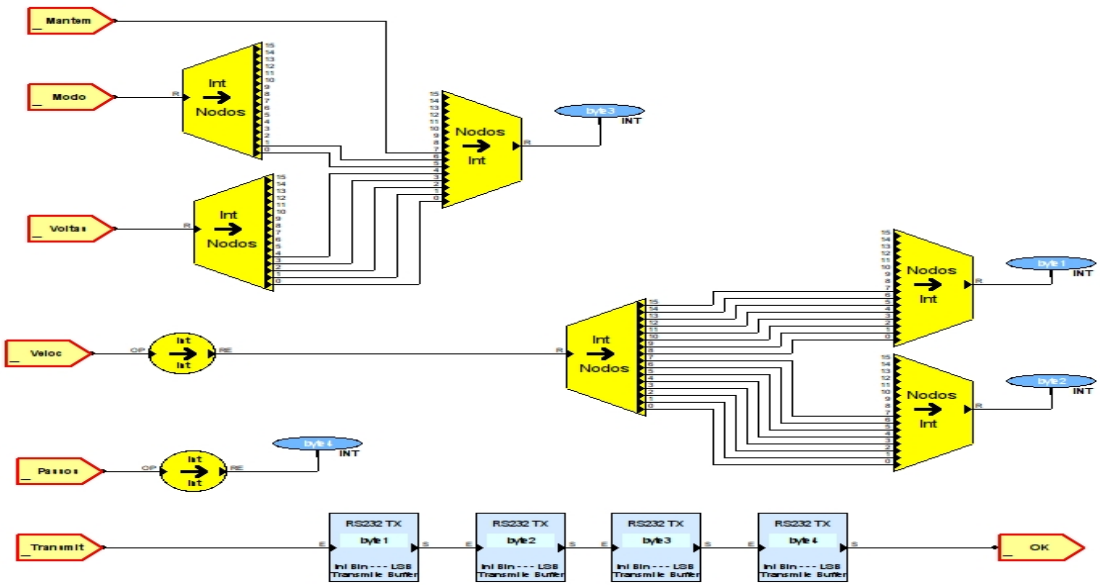


Figura 23 Macro.

A parte do código que programa o *byte* 3 do *frame,* utilizando a macro acima, é exemplificada na Figura 24.

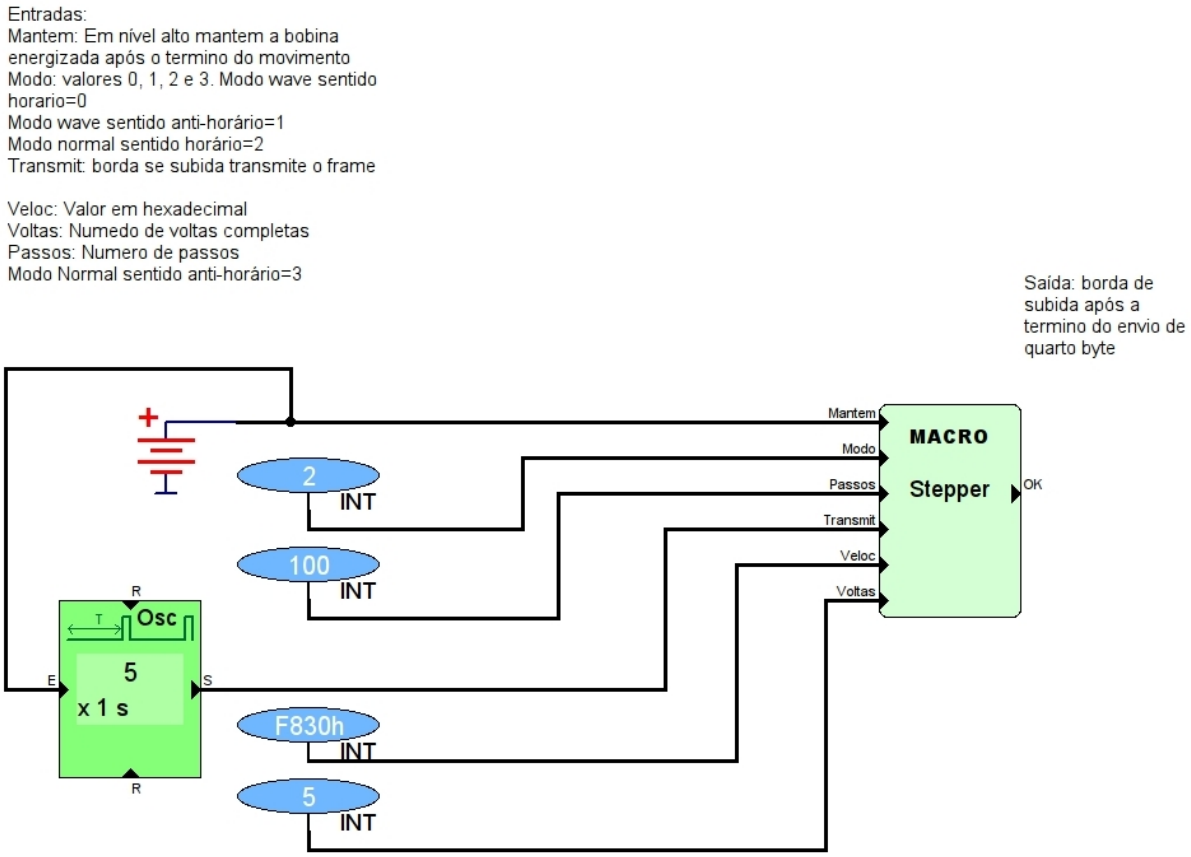


Figura 24 Exemplo programação.

6 Resultados obtidos

Para velocidades baixas e movimentos curtos, esse projeto atende aplicações de maneira que pode-se considerar satisfatória pelo fato que não foi observado nos testes que o motor apresentava perda de passos ou de sincronismo. A respeito da velocidade de rotação, os valores não foram medidos, apenas calculados de acordo com a frequência dos pulsos, logo um estudo mais completo exigiria um sistema de medida para então sim comparar-se os resultados e as diferenças entre teoria e prática. Os modos implementados foram os dois tipo de acionamento *full step.* Algumas particularidades que merecem destaque é que não há acúmulo nem fila de instruções do motor na parte do microcontrolador. A cada novo comando uma nova sequência de *bytes* deve ser enviada pelo CLP. Isso não apresenta uma limitação prática, pois o CLP precisa de aproximadamente 1 ms para enviar um novo *frame*. Para movimentos contínuos maiores que 32 voltas, podemos sobrescrever com um novo comando antes de terminar as 32 voltas, e assim dando continuidade ao movimento.

Em comandos para o CLP manter uma posição estática com torque (bobinas energizadas), uma ordem de movimento com 0 voltas e 0 passos, e com o oitavo bit do *byte* 3 no valor 1 pode ser enviada. O tempo de permanecia em alguma posição ou o tempo para o novo movimento é definido pela programação do CLP.

O sistema, ao ser inicializado, começa em um estado de desenergização das bobinas. Ao receber um *frame* pela serial, ele escreverá o valor inicial de 0001 e esse primeiro passo pode ser no sentido do movimento desejado, ou no sentido contrário ou o motor fica parado, isso depende da posição em que o motor parou no último movimento. Independentemente disso, os passos só irão começar a contar após o motor entrar na posição inicial forçada pelo valor 0001. Logo, não há perda de passos no comando a ser executado.

A implementação realizada com os transistores TIP 120 mostrou-se eficiente, embora os resistores de coletor apresentem um aquecimento elevado. A frequência máxima de chaveamento obtida foi de 666 Hz, que corresponde a uma velocidade de 150 rpm. Nessa faixa de frequência, esse transistor não apresenta perda de desempenho.

A Tabela 10 mostra alguns valores que devem ser enviados pelo CLP para obter frequência de rotações por minuto indicada na coluna 2.

Tabela 10 Valores *byte* 1 e 2.

valor byte 1	valor byte 2	rpm
248	48	150,0
244	72	100,0
240	96	75,0
236	120	60,0
232	144	50,0
228	168	42,9
224	192	37,5
220	216	33,3
216	240	30,0
213	8	27,3
209	32	25,0
205	56	23,1
201	80	21,4
197	104	20,0
193	128	18,8
189	152	17,6
185	176	16,7
181	200	15,8
177	224	15,0

Tabela 11 valores *byte* 3.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valor decimal	
0	0	1	0	1	0	1	0	42	Modo wave, sentido anti-horário, fase desenergizada
1	0	0	0	1	0	1	0	138	Modo wave, sentido horário, fase energizada

7 Conclusões

Conseguiu-se, ao fim do projeto, um sistema funcional de acionamento de motor de passo, com uma programação por diagramas de blocos de fácil aprendizado e implementação, e que pode já atender a várias aplicações. Algumas melhorias podem ser acrescentadas como controle no modo de *half step* e *microstepping*, além de um modo capaz de alimentar motores bipolares, com inversão do sentido da corrente no motor.

Um produto comercial desse equipamento precisaria de desenvolvimento em várias áreas que fogem do foco desse trabalho. Poderia se incluir um fonte interna, especificações e limites de operação, normatização, composição de manuais, testes, confiabilidade, geração de ruído, sistemas de *reset* de posição, previsão de vida útil, valor de mercado, análise de produtos concorrentes, entre outros.

Contudo, a parte que teve maior destaque no trabalho foi o desenvolvimento da comunicação com o CLP, codificação dos comandos e principalmente a programação do microcontrolador, onde foi criado todo o código do zero e desenvolvido um algoritmo próprio. É verdade que pode-se desenvolver um código mais eficiente e mais rápido, porém, para um microcontrolador com um *clock* de 12 MHz, alguns microssegundos a mais não apresentarão perda de desempenho para essa aplicação.

BIBLIOGRAFIA

[1] KENJO, Takashi; SUGAWARA, Akira. **Stepping Motors:** And their microprocessor controls. 2. ed. New York: Oxford University Press, 1994. 296 p

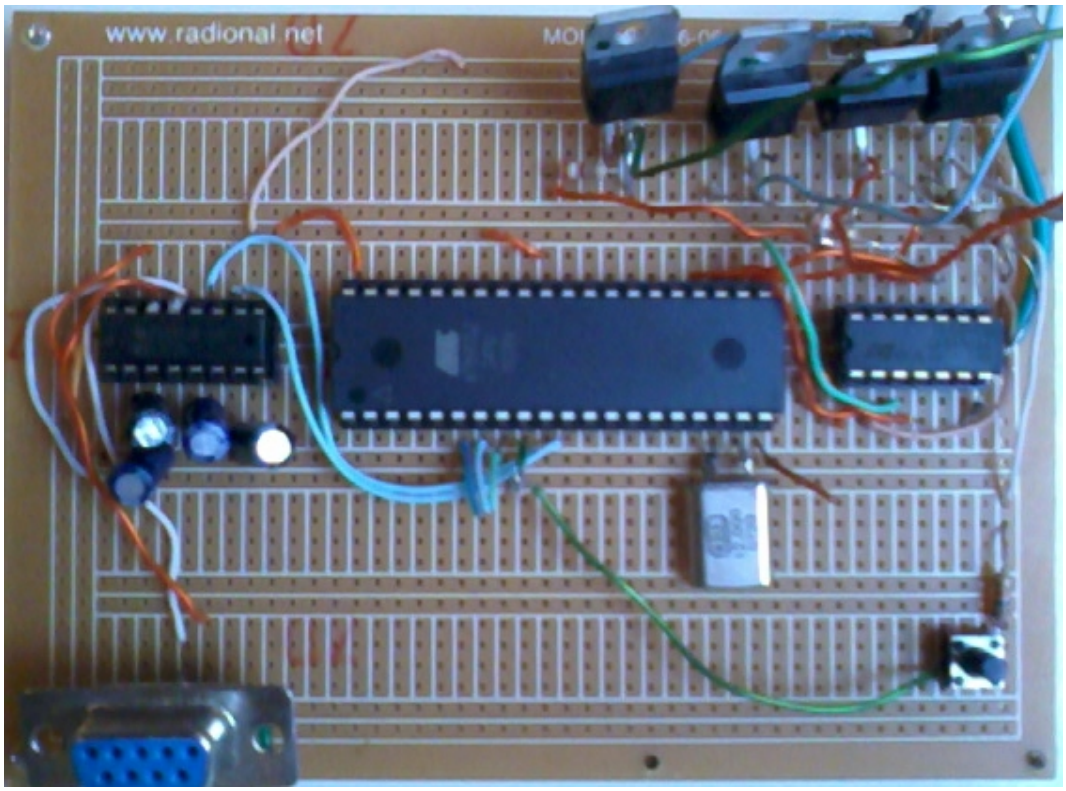
[2] CALCUTT, David; COWAN, Fred; PARCHIZADEH, Hassan. 8051 **Microcontrollers:** An Applications-Based Introduction. 2. ed. New York: Elsevier, 2004. 417 p.

[3] AXELSON, Jan. **Serial Port Complete:** Programming and Circuits for RS-232 and RS-485 links and networks. Madison: Lakeview Research, 2000. 321 p.

[4] SEDRA, Adel S.; SMITH, Kenneth C. **Microelectronic Circuits.** 6. ed. New York: Oxford University Press, 2010. 1456 p.

ANEXO A

Foto do circuito em placa perfurada



Motor utilizado no projeto

ANEXO B

Código em Assembly

```
BAUDL EQU 0F6H ; valor para gerar baudrate de 38400bps
BAUDH EQU 0FFH ;valor para gerar baudrate de 38400bps
VERMELHO EQU 01H ; valor para acionamento da fase do fio vermelho do conector do motor
LARANJA EQU 02H
AMARELO EQU 04H
AZUL EQU 08H


ORG 0000   ; inicio do codigo
JMP INICIO ; pula os endereços reservados ao atendimento das interrupções

ORG 000BH  ;endereço de atendimento de interrupção do timer 0
jmp timer_int  ; pula para atendimento da interrupção do timer 0

ORG 0023H  ; endereço de atendimento da interrupção da UART
LJMP SERIAL  ; pula para atendimento da interrupção da porta serial

INICIO:  ; inicio do programa

mov r0, #0h   ; reseta os valores dos registradores responsáveis pelo movimento do motor
mov r1, #0h
mov r2, #0h
mov r3, #0h
mov r4, #0h
mov p2, #0h

inicia_serial:    ; inicialização da porta serial

MOV SCON, #01010000B  ; definição do modo de funcionamento da porta serial
MOV RCAP2L, #BAUDL    ; valor de recarregamento automatico para gerar a baud rate
MOV RCAP2H, #BAUDH    ; valor de recarregamento automatico para gerar a baud rate
```

```
MOV T2CON, #00110100B  ; definição do modo de funcionamento do timer 2, responsavel pela temporização da baudrate
SETB ES   ; ativa interrupção da porta serial

inicia_timer0:
MOV TMOD, #01d ;timer 0 MODE 1 - 16 BITS
SETB TR0      ; liga timer 0
SETB ET0      ; ativa interrupção do timer 0
SETB EA       ; ativa todas as interrupções

loop:    ; loop infinito de aguardo das interrupções

JMP loop

timer_int:     ; atendimento da interrupção do timer 0
MOV TH0, r0;   ; carrega os valores iniciais do timer 0, responsáveis pela velocidade do motor
MOV TL0, r1    ; carrega os valores iniciais do timer 0, responsáveis pela velocidade do motor

mov a, r4  ; move o valor do numero de passos para acc
jz r4zero  ; testa para saber se há passos a serem executatos
jmp gira   ; se R4 /= 0 motor deve girar

r4zero:
mov a, r3
jz retit0  ; se tanto r4 quanto r3 forem zero, não movimenta o motor e volta da interrupção
dec r3     ; se r3 não for zero, então existe voltas completas a cumprir, e decrementa uma volta
mov r4, #200d ; move a quantidade de passos de uma volta completa para o registrados r4

gira:  ; inicia sequencia de comandos para saber como será feito o próximo passo
dec r4
;ljmpwcd = long jump modo wave sentido horario
jb 20h, ljmpwcw    ; testa os bits do modo de acionamento e pula para Long jump wave clockwise
jb 21h, ljmpwccw   ; testa os bits do modo de acionamento
jb 22h, ljmpncw    ; testa os bits do modo de acionamento
jb 23h, ljmpnccw   ; testa os bits do modo de acionamento
```

```
; obs: a instrução jumpbit faz um pulo curto, logo um pulo mais longo precisa de uma etapa intermediária


retit0: ; se não há movimento a cumprir
jb 24h, retii  ; se o bit 24h está setado, significa que o motor fica energizado, logo volta da interrupção sem alterar o valor de p2
mov p2, #00h  ; se 24h é zero, então motor fica desenergizado e  move zero pra p2
retii:
reti


ljmpwcw:  ; pulo intermediário pois a instrução jb não alcança endereços maiores que 8 bits na memória de programa
nop
nop
nop
nop
nop
nop
ljmp wcw


ljmpwccw:
nop
nop
nop
nop
ljmp wccw


ljmpncw:
nop
nop
ljmp ncw


ljmpnccw:
ljmp nccw


SERIAL:
```

```
clr ri   ; limpa bit ri
mov r0, sbuf ; move o byte recebido para o registrador THO
jnb ri, $    ;espera próximo byte
mov r1, sbuf  ; move o byte recebido para o registrador TL0
clr ri
jnb ri, $    ; espera ri
mov r2, sbuf  ; move o terceiro byte para seu registrador r2
clr ri
jnb ri, $
mov r4, sbuf  ; move o quarto byte para seu registrador

clr ri
mov a, r2
anl a, #00011111b ; E lógico entre o valor de acc e o valor 31d para eliminar os 3 bits mais significativos de R2
mov r3, a ; r3 recebe o numero de voltas completas depois de fazer a operação and com o valor 00011111b
call terceirobyte ; chama rotina que decodifica modos de acionamento e seta suas respectivas flags
reti; retorna do atendimento da interrupção da serial

terceirobyte: ; rotina de tratamento do byte 3
clr 20h  ; limpa todos os bits dos modos de acionamento
clr 21h
clr 22h
clr 23h
clr 24h
mov a, r2; move o valor de r2 para o acumulador para iniciar sequencia de rotação a esquerda com carry
clr c ; reseta o valor do carry
rlc a; rotaciona acc  para esquerda com carry para decodificar os modos de operação
jc set24h ; se há bit no primeiro carry, seta 24h

volta: ; continua rotações do acc
clr c ; reseta o valor do carry
rlc a; rotaciona acc  para esquerda com carry
jc carry1  ; desvia se há carry
```

```
rlc a; rotaciona acc
jc carry2  ; desvia se há carry
setb 20h ; se não há carry nas duas rotações, desvia para WCW,
ret

carry1:  ; carry presente para decodificar o modo de operação
clr c
rlc a
jc set23h ; dois carry seguidos representa cod 11 pula para nccw
setb 22h

ret
carry2:
clr c
setb 21h

set23h:
setb 23h
ret



set24h:
setb 24h
sjmp volta

WCCW:   ;modo wave counterclockwise(anti horario))


jb P2.0, ESTADO2a ; testa o estado da p2 e desvia para o proximo estado
jb P2.1, ESTADO3a ; os estados "a" e b" são para diferenciar do movimento horário do movimento anti-horário
jb P2.2, ESTADO4a
jb P2.3, ESTADO1a
mov p2, #01h   ; move valor inicial de P2 caso o motor esteja parado e desenergizado

WCW: ; modo wave clockwise
```

```
jb P2.0, ESTADO4b
jb P2.3, ESTADO3b
jb P2.2, ESTADO2b
jb P2.1, ESTADO1b
mov p2, #01h


    ESTADO1a:
    MOV P2, #vermelho
    RETI


    ESTADO2a:
    nop  ; instruções para gastar tempo por causa dos da ordem dos jump bit das flags de modo de operação.
    nop
    nop
    nop
    nop
    nop

    MOV P2, #laranja
    RETI

    ESTADO3a:
    nop
    nop
    nop
    nop

    MOV P2, #amarelo
    RETI

    ESTADO4a:
    nop
```

```
nop
MOV P2, #azul
RETI

 ESTADO1b:
MOV P2, #vermelho
RETI

ESTADO2b:
nop
nop

MOV P2, #laranja
RETI

ESTADO3b:
nop
nop
nop
nop

MOV P2, #amarelo
RETI


ESTADO4b:
nop
nop
nop
nop
nop
nop

MOV P2, #azul
RETI
```

```
nccw:
jb p2.0, tp2_0  ; quando executando modo normal, cada estado da porta P2 depende de 2 bits, logo dois bits tem que serem testados nas sub-rotinas tp2_1, tp2_2, etc
jb p2.1, tp2_1  ; test p2.1
jb p2.2, tp2_2  ; test p2.2
jb p2.3, tp2_3
mov p2, #01h

tp2_0:   ; se há bit com valor 1 em p2.0
nop
nop
nop
nop
nop
nop
jb p2.3, novecw ; nove cw significa o valor nove escrito na porta p2 sendo executado o modo normal
jmp trescw

tp2_1:
nop
nop
nop
nop
jb p2.0, trescw
jmp seiscw

tp2_2:
nop
nop
jb p2.1, seiscw
jmp dozecw

tp2_3:
jb p2.2, dozecw
jmp novecw
```

```
novecw:
mov p2, #03h
reti
trescw:
mov p2, #06h
reti
seiscw:
mov p2, #0ch
reti
dozecw:
mov p2, #09h
reti
ncw:
jb p2.0, tp2_0b
jb p2.1, tp2_1b
jb p2.2, tp2_2b
jb p2.3, tp2_3b
mov p2, #01h
tp2_0b:
nop
nop
nop
nop
nop
nop
jb p2.3, noveccw
jmp tresccw
tp2_1b:
nop
nop
nop
nop
jb p2.0, tresccw
jmp seisccw
```

```
tp2_2b:
nop
nop
jb p2.1, seisccw
jmp dozeccw

tp2_3b:
jb p2.2, dozeccw
jmp noveccw

noveccw:
mov p2, #0ch
reti
tresccw:
mov p2, #09h
reti
seisccw:
mov p2, #03h
reti
dozeccw:
mov p2, #06h
reti
end
```