TUTORIAL
CONTROLE DE
MOTOR DE PASSO

Autor: Luís Fernando Patsko
Nível: Intermediário
Criação: 22/02/2006
Última versão: 18/12/2006

Maxwell Bohr
Instrumentação Eletrônica

http://www.maxwellbohr.com.br
contato@maxwellbohr.com.br

PdP
Pesquisa e Desenvolvimento de Produtos

http://www.automato.com.br
atendimento@automato.com.br

1 – Introdução

Esse tutorial abordará a confecção de uma placa para o controle de um motor de passo e a programação para que se possa controlá-lo por meio das portas de saídas digitais do MEC1000 ou do KDR5000. O motor de passo é muito utilizado na robótica porque pode ter sua rotação controlada por meio de passos, como o nome já diz, e isso é indispensável em alguns aparelhos onde é exigido um maior grau de precisão, pois o uso de motores de corrente contínua convencionais poderia resultar em danos no aparelho ou em mal funcionamento.

2 – Material

O material necessário será um motor de passo, um integrado ULN2003, um conector latch de 10 vias macho, um conector latch de 10 vias fêmea, um cabo flat e uma placa de fenolite.

3 – Funcionamento

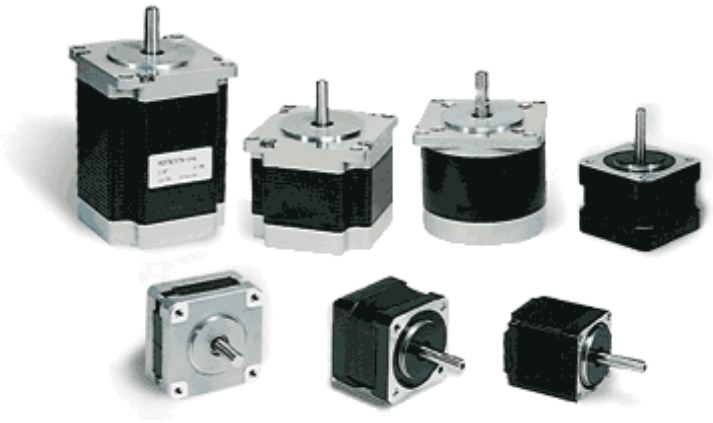


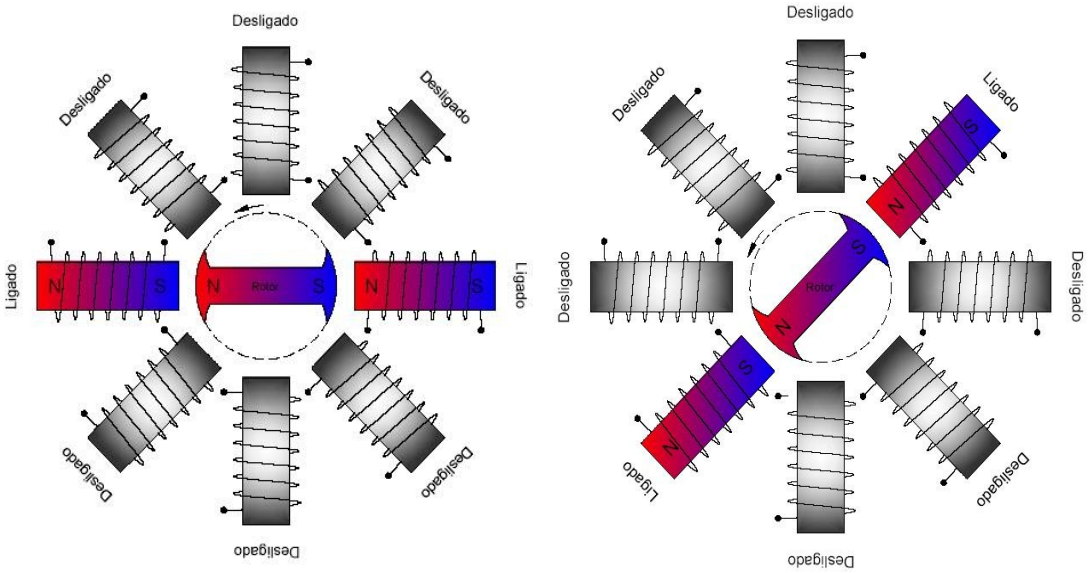
Figura 1. Motores de passo

Os motores elétricos são dispositivos eletromecânicos que transformam energia elétrica em movimento rotativo por meio de ímãs e indutores em seu interior. Os modelos de motores de aplicação mais difundida são os motores de corrente contínua (CC ou DC), os de corrente alternada (CA ou AC), os universais, que funcionam tanto com corrente contínua quanto alternada, os servomotores e os de passo.

O motor de passo é um motor com uma precisão muito grande do seu movimento. São utilizados onde é necessário o controle do número de rotações é muito importante, tais como em impressoras, drives de disquete e sistemas de automação industrial e robótica, pois, se não houvesse esse controle, o movimento contínuo poderia estragá-los. O passo que esse motor pode dar é o menor deslocamento angular para o qual está projetado. O número de passos destes motores dependem do número de pólos que seu rotor possui. Assim, existem diversas resoluções para eles como, por exemplo, 0.72, 1.8, 3.6, 7.5, 15 e até 90 graus, ou seja, 500, 200, 100, 48, 24 e 4 passos

por rotação, respectivamente. Além do mais, hoje existem controladores de modulação de largura de pulso onde se obtém micropassos, desse modo observa-se uma resolução de posicionamento infinitamente preciso. Para se ter uma idéia, há controladores que operam na faixa de 50.000 micropassos por rotação.

No seu interior há estatores formados por bobinas que geram, quando percorridos por uma corrente elétrica, o campo magnético necessário para o movimento do rotor, que é construído com ímas permanentes confeccionados de acordo com o número de passos. Essa rotação é controlada por meio de um circuito externo que promove a oscilação do sinal que percorrerá os pares de estatores e, por isso, não pode ser conectado diretamente à alimentação, pois desse modo não haveria a pulsação necessária para que o motor possa girar.

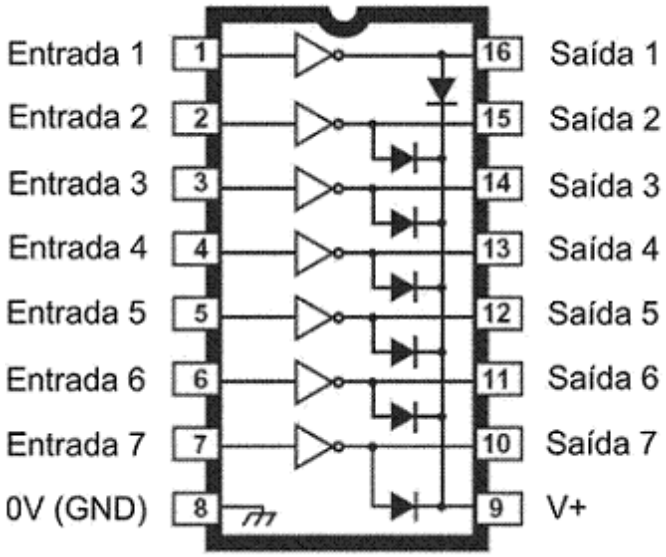


O movimento se dá quando há um ciclo no campo magnético dos estatores, ou seja, a cada vez que um par de estatores é magnetizado, ocorre um passo do eixo do motor. É necessário construir o oscilador para o motor de passo corretamente, conforme o número de passos que se queira adquirir. No nosso caso, estaremos utilizando um motor com 4 pares de estatores e um rotor com 12 pólos. Desse modo, estaremos trabalhando com um motor que dá 48 passos por rotação ou um passo a cada 7,5º.

Para que possamos controlar o motor precisamos saber para qual tensão ele está projetado e qual a corrente que ele consome. No nosso caso estaremos alimentando o motor com uma tensão de 12 V e pela "Lei de Ohm" poderemos calcular a corrente que será consumida pelo motor. Observando-se os dados do motor descobrimos que sua resistência interna é de 135Ω, então, fazendo os cálculos descobrimos a corrente que ele consumirá:

$$V = R \cdot i$$
$$12 = 135 \cdot i$$
$$i = \frac{12}{135}$$
$$i \approx 0{,}09A = 90mA.$$

Desse modo poderemos estar utilizando o ULN2003, que nada mais é do que um *array de transistores Darlington*, que trabalha com tensões de até 50V e suporta uma corrente de até 500mA, o suficiente para o controle do nosso motor de passo. Este integrado possui 7 entradas para o controle de até 7 saídas que serão ligadas às saídas dos estatores. É necessário em primeiro lugar descobrir qual a saída comum aos estatores e esta será ligada à entrada de 12V. As conexões das saídas dos estatores serão ligados aos pinos 16, 15, 14 e 13 do integrado e as saídas digitais do Kit correspondentes a eles serão conectadas aos pinos 1, 2, 3 e 4. O integrado estará ligado à alimentação por meio dos pinos 8 (GND) e 9 (V+). As portas 1, 2, 3 e 4 serão ligados por meio de cabos flat ao MEC1000 ou ao KDR5000.



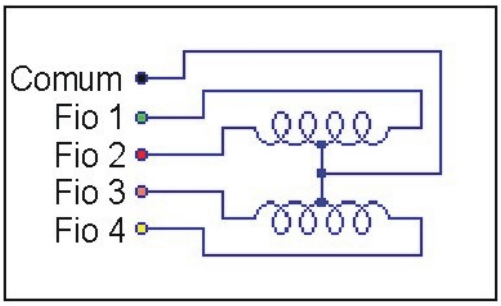
ULN2003

Há três modos básicos de operação do motor de passo: dois onde se usa o passo completo, um com apenas um estator energizado por passo, onde o consumo de energia e torque são baixos, e o outro com dois estatores energizados por passo, onde torque e consumo de energia são maiores; e um de meio-passo, gerando um torque próximo ao de passo completo com dois estatores energizados, com um consumo superior de energia, porém com uma precisão maior deslocamento.

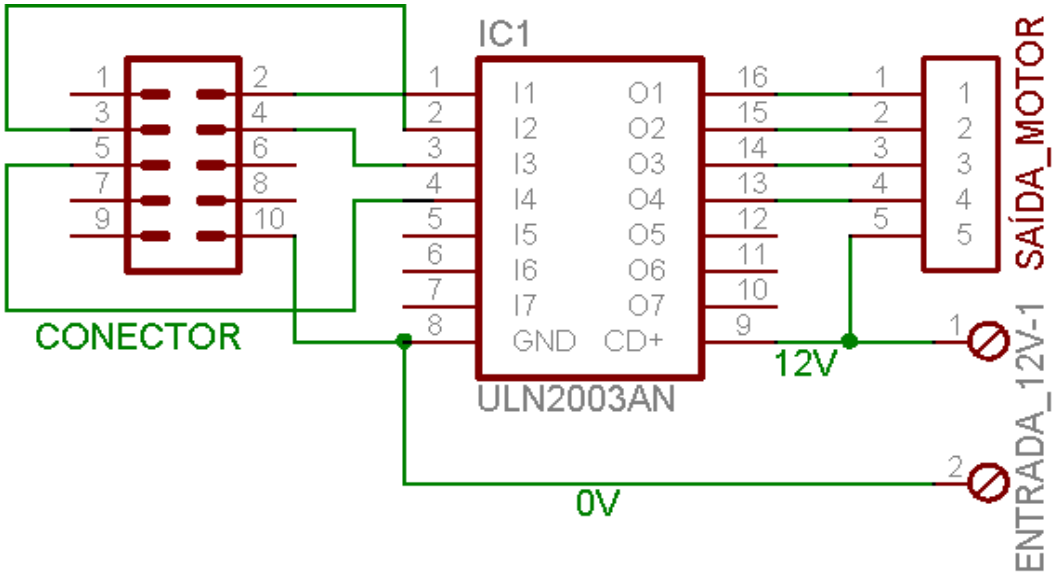
Com a finalidade de uma explicação mais clara e uma linha de código menor, o programa de controle do motor de passo disponibilizará apenas o uso das operações de passo completo.

4 – Montagem

Para a montagem da placa para o motor de passo é necessário observar bem a polaridade dos fios para que não haja erro na hora da confecção da placa. O fio comum aos estatores é aquele cuja resistência quando medida entre dois fios dá o valor nominal da resistência do motor. Este fio deve ser ligado à alimentação de 12V. Se medirmos a resistência entre dois fios que são os extremos da bobina, esta será o dobro da resistência nominal do motor.

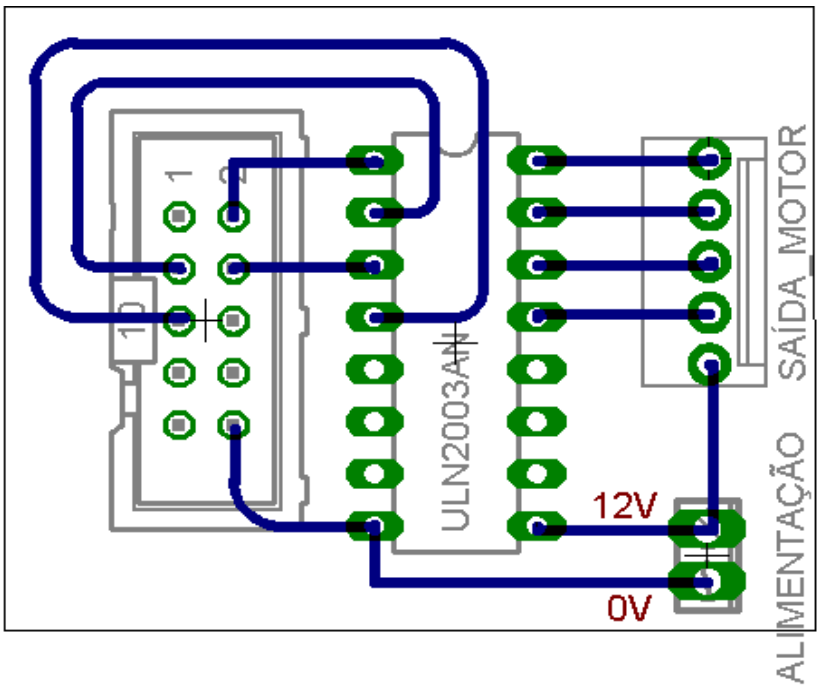


Começaremos pelo diagrama esquemático do circuito para que não haja erro na hora de confeccionarmos a placa. Para isso temos que ter à mão o integrado ULN2003 pois este é o único componente eletrônico que será soldado à placa. O esquemático está mostrado a seguir.



Observe que tanto o ULN2003 como o próprio motor de passo não utilizarão o pino de alimentação de 5 Volts disponível no pino 1 das portas de saídas digitais. Eles deverão ser alimentados com uma tensão de 12 Volts, disponível nas saídas de alimentação do Kit. Sendo assim é possível soldar um borne de 2 vias ou mesmo soldar dois fios diretamente na placa, sendo que um será o terminal positivo e o outro será o terminal negativo. Basta ligá-los nas saídas de alimentação correspondentes.

O desenho da placa, vista pelo lado dos componentes, é mostrado a seguir.



Devemos transferir o desenho para o lado cobreado da placa de fenolite tomando o cuidado de invertê-lo. Depois disso é só fazer a perfuração da placa e desenhar as trilhas até às saídas dos cabos flat. Observe que utilizaremos apenas 4 entradas e saídas do integrado, preferencialmente a dos pinos 1, 2, 3 e 4 como entradas, os pinos 16, 15, 14 e 13 respectivamente como saídas das entradas utilizadas e os pinos de alimentação que são o 8 (GND) e 9 (V+), podendo ligar o terminal comum do motor de passo com este último pino, pois ambos vão ao positivo. Após o desenho das trilhas colocamos a placa para ser corroída no percloreto de ferro.

Depois de feita a corrosão da placa, soldaremos o ULN2003 conforme o diagrama, observando muito bem sua pinagem, pois caso seja invertido, ele poderá ser danificado. Com muita atenção, compare a posição em que será soldado com o esquema e com a pinagem do componente, para conferir se os pinos estarão conectados adequadamente. Soldaremos também um conector latch, com o propósito de ligar um cabo flat ao Kit e o conector adequado ao motor de passo. Para as entradas de alimentação, podemos soldar um borne de 2 vias ou então soldar dois fios diretamente à placa.

5 – Programação

Depois da criação da placa para o controle do motor de passo, é necessário criar um programa específico para controlá-lo. Isso é necessário devido ao próprio funcionamento do motor. Como vimos logo acima, não basta apenas alimentá-lo corretamente, é necessário haver uma oscilação que faça com que os estatores proporcionem o movimento. Devemos então criar um programa que proporcione essa oscilação nas saídas digitais.

Figura 2. Programa de controle da placa de motor de passo.

Nosso primeiro passo é criar a interface gráfica do programa. Vamos utilizar o projeto criado no tutorial Base que já nos fornece algumas funcionalidades interessantes. Para isso copiamos o projeto daquele tutorial e em cima dele vamos adicionar alguns componentes gráficos extras.

A interface gráfica é composta por dois SpeedButton, dois Labels e dois ComboBox. Os botões serão destinados a ligar e desligar o motor nos modos de torque alto e torque baixo. Uma ComboBox selecionará a porta de saída digital e a outra o sentido da rotação do motor e os Labels apenas indicarão o conteúdo de cada ComboBox. Também existe um componente Timer, que não é visível durante a execução do programa, mas é essencial para seu funcionamento.

O primeiro passo é modificar a propriedade Caption do Form principal de "Projeto Base" para "Motor de passo – placa". É importante deixar o nome desse programa um pouco diferente do que é destinado a controlar o motor de passo do Kit, para evitar confusões. Em seguida adicionaremos um Label e um ComboBox para a seleção da porta de saídas digitais na interface. Esses componentes podem ser encontrados na aba "Standard".

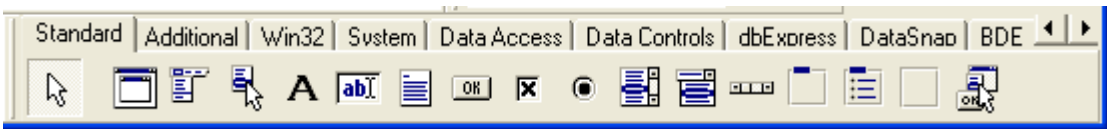


Figura 3: *Aba "Standard" da Barra de componente.*

O componente Label possui o seguinte ícone.

Figura 4: *Ícone do componente Label.*

E o componente ComboBox possui o seguinte ícone.

Figura 5: *Ícone do componente ComboBox.*

Vamos então adicioná-los e modificar as propriedades do Label.

Name	=	LabelPorta
Caption	=	Porta:
Font/Style/fsBold	=	true

Em seguida, modificaremos as propriedades do ComboBox.

Name	=	ComboBoxPorta
Style	=	csDropDownList
Items.Strings	=	Porta 0, Porta 1
ItemIndex	=	0

Nesse momento, o nosso Form deverá ter a seguinte aparência:

Figura 6: *Form após inclusão do Label e do ComboBox.*

O próximo passo será adicionar os componentes responsáveis pela seleção do sentido do motor. Vamos adicionar um Label e modificar suas propriedades.

Name	=	LabelSentido
Caption	=	Sentido:
Font/Style/fsBold	=	true

Em seguida, adicionaremos um ComboBox e modificaremos as seguintes propriedades.

Name	=	ComboBoxSentido
Style	=	csDropDownList
Items.Strings	=	Horário, Anti-horário
ItemIndex	=	0

Nosso Form então deverá se parecer com o seguinte:

Figura 7: *Form após inclusão do Label e do ComboBox.*

Devemos agora adicionar os SpeedButtons, cuja função será acionar ou desligar o motor. Como temos dois modos de operação possíveis, devemos colocar dois SpeedButtons, que encontram-se na aba de componentes "Additional".

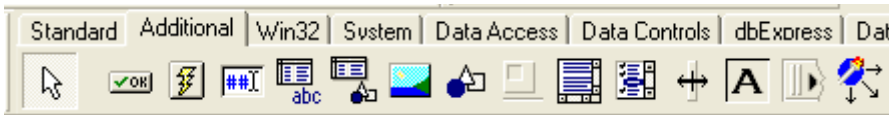


Figura 8: *Aba "Additional" da barra de componentes.*

O componente SpeedButton possui o seguinte ícone nessa barra.

Figura 9: *Ícone do componente Button.*

Name	=	SpeedButtonBaixo
Caption	=	Torque Baixo
Font/Style/fsBold	=	true
GroupIndex	=	1
AllowAllUp	=	True

Name	=	SpeedButtonAlto
Caption	=	Torque Alto
Font/Style/fsBold	=	true
GroupIndex	=	2
AllowAllUp	=	True

O último componente a ser adicionado é um Timer, que não será visível na interface gráfica do programa, no entanto, é essencial para o seu funcionamento. O componente Timer encontra-se na aba "System" da barra de componentes. A seguir uma imagem dessa aba.

Figura 10: *Aba "System" da barra de componentes.*

O componente Timer possui o seguinte ícone:

Figura 11: *Ícone do componente Timer.*

Devemos adicioná-lo e modificar as seguintes propriedades:

Name	=	TimerMotor
Interval	=	400

O nosso Form então ficará da seguinte maneira:

Figura 12: *Form após inclusão dos SpeedButtons e do Timer.*

Com a inclusão desse componente, encerramos a criação da parte gráfica e podemos partir

para a implementação do código do programa. É importante ressaltar que é necessária muita atenção nesse momento. Como a linha de código é relativamente extensa, é fácil ocorrerem pequenos deslizes que impeçam que o programa funcione.

Como já foi mencionado, é necessário criar um programa que acione uma saída digital por vez, possibilitando o movimento do eixo do motor. A cada saída acionada, uma porta do integrado ULN2003 será acionada que, por sua vez, acionará um estator do motor. Quando um estator do motor é acionado, seu eixo dará um passo. Para disponibilizar um maior torque do motor, basta apenas acionarmos duas saídas digitais a cada vez, que ligarão dois estatores simultaneamente.

O primeiro passo para a implementação do código será dado no manipulador do evento OnTimer, do TimerMotor que adicionamos no Form. Para criar esse manipulador podemos selecionar o componente, ir no **Object Inspector**, selecionar a aba Events e dar um duplo clique sobre a linha que está escrito OnTimer. Uma forma mais fácil de isso é apenas dar um duplo clique sobre o timer no Form e com isso o Delphi irá criar automaticamente um manipulador para o evento OnTimer.

```
procedure TFormMain.TimerCheckTimer(Sender: TObject);
begin
end;
```

Como esse componente timer será responsável por enviar continuamente para o Kit as informações sobre o modo de operação, porta utilizada e sentido, devemos colocar um bloco **try-except** para evitar que o programa trave caso não haja nenhum Kit conectado.

```
procedure TFormMain.TimerCheckTimer(Sender: TObject);
begin
  try
  begin

  end;
  except
  end;
end;
```

O próximo passo será então implementar o código para os botões que acionarão o motor. No mesmo procedimento, vamos utilizar uma estrutura de repetição **while** e uma estrutura de decisão **if** para cada botão.

```
procedure TFormMain.TimerCheckTimer(Sender: TObject);
begin
  try
```

```
    begin
       while SpeedButtonBaixo.Down = true do
         begin
         end;
       if  SpeedButtonBaixo.Down = false do
         begin
         end;
       while SpeedButtonAlto.Down = true do
         begin
         end;
       if SpeedButtonAlto.Down = false do
         begin
         end;
    end;
    except
    end;
end;
```

Agora, faremos uma pausa para analisar os comandos que serão enviados pelas saídas digitais. Utilizaremos o método DigitalPortWrite para enviar os comandos para a placa de controle de motores-de-passo. Esse método possui a seguinte declaração:

```
Procedure DigitalPortWrite(port : Integer;
                                        value : Byte);
```

Ele possui dois parâmetros, um que indica a porta que queremos definir o estado e outro que define o estado da porta. O *Módulo de Entradas, Saídas e Servo-Motores* possui duas portas de saídas digitais, logo o primeiro parâmetro pode ser "0", para indicar a primeira porta, ou "1", para indicar a segunda. Esse parâmetro será selecionado pelo ComboBoxPorta.

Já o segundo parâmetro do método DigitalPortWrite define quais bits da porta deverão ser setados para o nível lógico 1 e quais deverão ser setados para o nível lógico 0. Esse parâmetro é um valor de 8 bits sendo que cada bit dele irá definir o estado de um bit da porta saída. O bit 0 da porta será representado pelo bit 0 do parâmetro, o bit 1 pelo bit 1 e assim por diante. Como o motor de passo utilizado possui 4 estatores, precisaremos setar apenas 4 bits.

Para acionar o primeiro estator, basta acionar a primeira saída digital da porta onde o motor está acoplado, utilizando o método DigitalPortWrite. O primeiro parâmetro será utilizado para selecionar a porta correta, enquanto que o segundo acionará as saídas desejadas. Enviando um número 1 no segundo parâmetro, acionaremos a primeira saída, pois esse número em código binário é 00000001. Caso queiramos acionar a terceira porta, devemos enviar o número 4, que em código

binário é 00000100. Caso, queiramos fazer com que um motor de passo gire, precisamos acionar seus estatores, um de cada vez. Podemos fazer isso enviando um número, cujo equivalente em binário seja adequado para acionar cada uma das 4 saídas durante um curto intervalo. A tabela a seguir mostra a relação entre o número enviado e seu valor binário.

Número decimal	**1**	**2**	**4**	**8**	**1**
Valor binário	**00000001**	**00000010**	**00000100**	**00001000**	**00000001**

Enviando esses valores, com um pequeno intervalo entre cada um, acionaremos cada uma das quatro primeiras saídas digitais, e consequentemente, os quatro estatores do motor de passo. Sendo assim, o motor se movimentará. Para fazer com que ele gire no sentido contrário, basta apenas inverter os números a serem enviados para a porta de saída digital.

No modo descrito acima, acionamos um estator de cada vez, utilizando o modo de baixo torque. Para um maior torque, basta apenas acionarmos dois estatores simultaneamente. Para isso, devemos modificar os valores a serem enviados. Ao enviar esses valores, com um pequeno intervalo entre cada um, teremos um maior torque disponível no motor de passo, embora o consumo também seja maior.

Número decimal	**3**	**6**	**12**	**9**	**3**
Valor binário	**00000011**	**00000110**	**00001100**	**00001001**	**00000011**

Agora já aprendemos como devemos acionar as saídas digitais de modo a proporcionar o giro do motor. Mas só há um problema. Precisamos que esses valores se repitam indefinidamente. Para isso utilizamos a estrutura **while**, cuja função é repetir inúmeras vezes uma determinada função. No nosso programa, enquanto um SpeedButton estiver acionado, ela enviará os bits para as saídas digitais infinitas vezes.

Dentro do bloco definido pela estrutura **while**, utilizaremos a estrutura de decisão **if-else**. Essa estrutura verificará o estado da ComboBox que indica o sentido e comparará com o valor 0. Se o item selecionado na ComboBox for igual a zero (que é o primeiro item), então será executada uma operação. Caso o item não seja igual a zero, então uma outra operação será executada, através da estrutura **else**.

Utilizaremos a função Sleep para gerar o intervalo entre cada estator acionado. Essa função simplesmente interrompe o andamento do código durante um intervalo, expresso em milésimos de segundo. No nosso programa, utilizaremos um intervalo de 100ms. Intervalos diferentes podem ser utilizados, mas um intervalo menor não é muito recomendado, pois o motor de passo não suporta velocidades de giro altas e poderá não funcionar.

Quando nenhum SpeedButton estiver acionado, o motor de passo deverá ser desligado. Para isso, basta utilizarmos duas estruturas de decisão **if**, sendo cada uma destinada a um dos botões. Então utilizaremos o método DigitalPortWrite, onde no seu segundo parâmetro colocaremos o número 0, que seta todas as saídas digitais da porta que está sendo utilizada em 0, desligando então o motor.

Desse modo, nossa linha de código ficará assim. Observe que, devido o tamanho da linha de código, é necessário prestar muita atenção e segui-la rigorosamente. Um pequeno erro de pontuação ou a falta de um **end** poderá fazer com que o programa não funcione.

```
procedure TFormMain.TimerMotorTimer(Sender: TObject);
begin
  try
  begin

       // Repetirá infinitamente a ação enquanto o
       // SpeedButtonBaixo estiver pressionado
       while SpeedButtonBaixo.Down = true do
       begin

           //verifica o sentido selecionado
           if ComboboxSentido.ItemIndex =0 then
           begin
              //atualiza o estado da porta de saída selecionada
              kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 1);

              //gera um intervalo de 100ms
              Sleep(100);

              kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 2);
              Sleep(100);
              kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 4);
              Sleep(100);
              kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 8);
              Sleep(100);
              Application.ProcessMessages();
           end
           else
           begin
              kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 8);
              Sleep(100);
              kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 4);
              Sleep(100);
              kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 2);
              Sleep(100);
              kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 1);
              Sleep(100);
              Application.ProcessMessages();
           end;
       end;

       // Realizará a instrução quando o SpeedButtonBaixo não
       // estiver acionado
       if SpeedButtonBaixo.Down = false then
       begin

           //seta todas as saídas digitais em zero
           kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 0);
       end;
```

```
        // Repetirá infinitamente a ação enquanto o SpeedButtonAlto
        // estiver pressionado
        while SpeedButtonAlto.Down = true do
        begin
            if ComboboxSentido.ItemIndex =0 then
            begin
               kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 3);
               Sleep(100);
               kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 6);
               Sleep(100);
               kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 12);
               Sleep(100);
               kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 9);
               Sleep(100);
               Application.ProcessMessages();
            end
            else
            begin
               kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 12);
               Sleep(100);
               kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 6);
               Sleep(100);
               kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 3);
               Sleep(100);
               kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 9);
               Sleep(100);
               Application.ProcessMessages();
            end;
        end;

        // Realizará a instrução quando o SpeedButtonAlto não
        // estiver acionado
        if SpeedButtonAlto.Down = false then
        begin
          kit.DigitalPortWrite(ComboBoxPorta.Itemindex, 0);
        end;

  end;
  except
  end;
end;

end.
```

Assim finalizamos a implementação do código do nosso programa e já podemos utilizá-lo. Uma boa maneira de realizar um primeiro teste com ele é ligar uma barra de LEDs (cuja montagem está descrita em outro tutorial). No modo de operação em baixo torque, um LED deve ser aceso por vez, sendo que ao chegar ao quarto LED, o ciclo deverá se reiniciar, e no modo de alto torque, dois LEDs. Para uma melhor visualisação, mude o tempo da funçao Sleep para 500ms, por exemplo.

Ao verificar que o programa funciona corretamente, basta apenas conectar a placa de controle com o ULN2003 na porta de saída digital desejada e conectar o motor de passo nesta. Também é necessário ligar a alimentação do motor de passo, ligando um cabo do terminal positivo e um do negativo nas saídas de alimentação de 12 Volts e 0 Volts (GND), presentes no Kit.

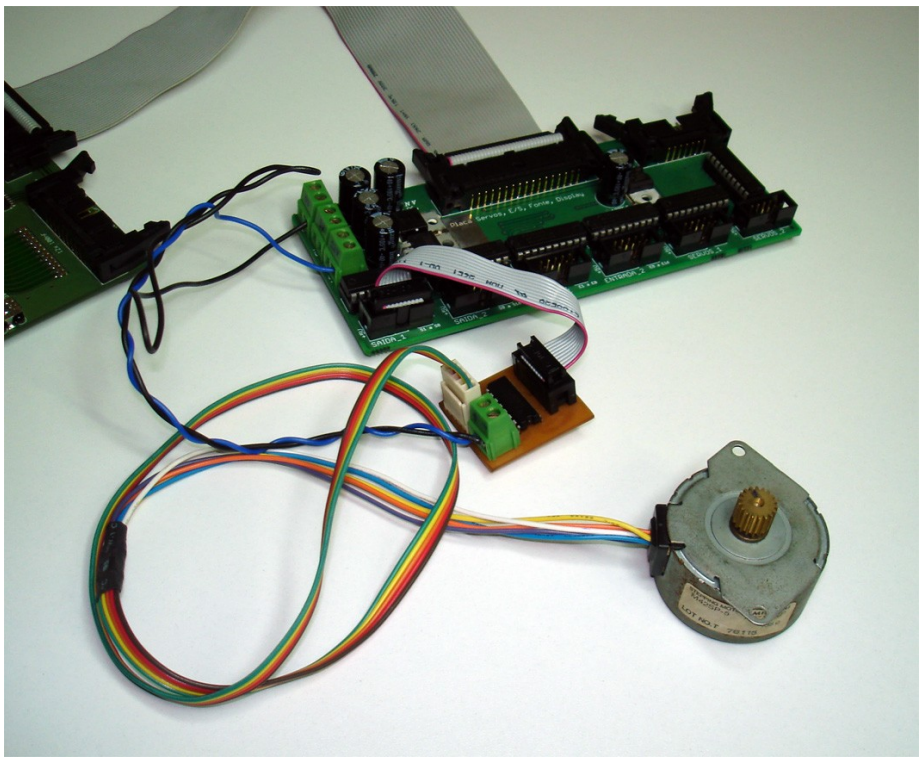


Figura 13. *Placa de controle de motor de passo ligado ao KDR5000.*

6 – Conclusão

Com esse tutorial, foi possível aprender a construção de uma placa destinada ao controle de um motor de passo utilizando um integrado ULN2003 e elaborar a programação destinada a controlá-lo, utilizando as portas de saídas digitais. Trata-se de um desenvolvimento complexo, mas que pode resultar em ótimos resultados, tanto em conhecimento adquirido quanto em praticidade de operação.