Alunos:

João Murilo Albuquerque Lourenco 364020

João Victor Lima Freitas Mota 364021

Marcos Horácio Borges Bressan 367459

Raísa Bruno Norões 367465

Suzane Carvalho de Sousa 368613

Professor: Dr. Luiz Henrique Silva Colado Barreto

Disciplina: TH0172 - MICROPROCESSADORES

Comunicação Serial e Controle de Motor de Passo com o Microcontrolador PIC16F628A

Relatório de Projeto

Sumário

[Introdução 3](#_Toc441673441)

[Sobre o projeto 3](#_Toc441673442)

[O Projeto 4](#_Toc441673443)

[Sobre o PIC16F628A 4](#_Toc441673444)

[*Universal Synchronous/Asynchronous Receiver/Transmitter* (USART) 7](#_Toc441673445)

[Tabela *ASCII* 9](#_Toc441673446)

[Materiais utilizados e Custo Projeto. 10](#_Toc441673447)

[*Display* de 7 segmentos 11](#_Toc441673448)

[Sobre o CI L7805CV 12](#_Toc441673449)

[Sobre o CI ULN2003 13](#_Toc441673450)

[Motor de Passo 15](#_Toc441673451)

[Acionamento do Motor 16](#_Toc441673452)

[PIC de Dados 17](#_Toc441673453)

[*Pinout* 17](#_Toc441673454)

[Programação 18](#_Toc441673455)

[Código – PIC de Dados 19](#_Toc441673456)

[PIC de Controle 23](#_Toc441673457)

[*Pinout* 23](#_Toc441673458)

[Constantes de programa 23](#_Toc441673459)

[Programação 24](#_Toc441673460)

[Código – PIC de Controle 27](#_Toc441673461)

[PCB – Placa de Circuito Impresso – PIC de Controle 31](#_Toc441673462)

[Esquemático do Projeto – Isis Proteus 32](#_Toc441673463)

[Conclusão 33](#_Toc441673464)

[Referências Bibliográficas 34](#_Toc441673465)

# Introdução

## Sobre o projeto

Para um melhor aprendizado prático do microcontrolador PIC16F628A, os professores da disciplina de Microprocessadores da Universidade Federal do Ceará exigiram que os alunos elaborassem um projeto.

               Esse projeto requer a utilização de dois microcontroladores, de um motor de passo, de um display de sete segmentos, de uma haste e de um círculo dividido igualmente em 16 setores – nos quais haverá a representação de caracteres diferentes em hexadecimal.

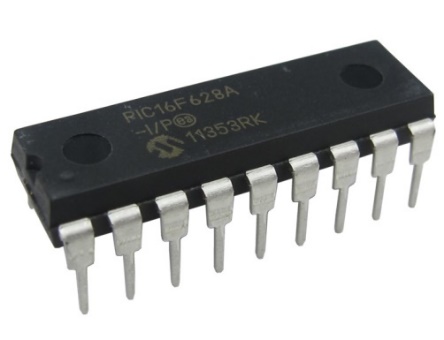
De modo geral, o projeto deverá funcionar da seguinte maneira: primeiramente, um microcontrolador será gravado com uma palavra – na qual cada caractere representa um código ASCII de 8 bits, ou seja, dois caracteres em hexadecimal. Posteriormente, por meio da transmissão serial, um segundo microcontrolador deverá receber esse conteúdo em hexadecimal, expondo-o sequencialmente digito a digito no círculo com o auxílio da haste. Para isso, o motor de passo é fundamental, pois este definirá a angulação feita pela haste de acordo com os seus passos, assim, esta apontará para um determinado caractere pertencente ao círculo.  Por fim, o display de sete segmentos deverá exibir o número ordinal correspondente à sequência dos caracteres na palavra em ASCII, o ponto do display acenderá a cada término da representação de um caractere em ASCII - os quais são representados por dois dígitos em hexadecimal.

# O Projeto

## Sobre o PIC16F628A

O PIC16F628A (Figura 1) é um microcontrolador – pequeno circuito integrado capaz de controlar processos lógicos utilizando-se de uma inteligência programável - produzido pela empresa Microchip Technology de expressiva participação no mercado de microcontroladores e semicondutores analógicos.

Figura - Imagem do microcontrolador PIC16F628A em encapsulamento DIP (*Dual In-line Package*).



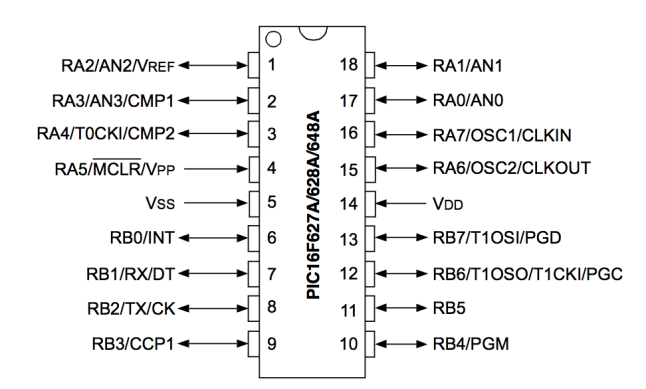
Possui Arquitetura Harvard para acesso aos dados e às instruções do programa, a qual é caracterizada por utilizar uma via de acesso para comunicação dos dados e outra diferente para comunicação com o programa. A maioria dos microcontroladores no mercado utiliza apenas uma, tanto para dados quanto para instruções do programa.

Utiliza também o padrão RISC (Reduced Instruction Set Computer, ou "computador com um conjunto reduzido de instruções"), que consiste num número limitado de instruções para facilitar o trabalho e baratear o custo.

Algumas características importantes sobre esse microcontrolador são listadas a seguir:

* Constituído por 18 pinos, 16 pinos os quais podem ser configurados como entrada e/ou saída;
* Possui 35 instruções;
* Permite um sinal de *clock* de frequência de até 20Mhz;
* Dados de 8 bits de endereço por memória;
* Memória de programa do tipo *Flash* de 2048 posições de 14 *bits*;
* Memória EEPROM de 128 localidades de 8 *bits*;
* 224 bytes de memória RAM para dados;

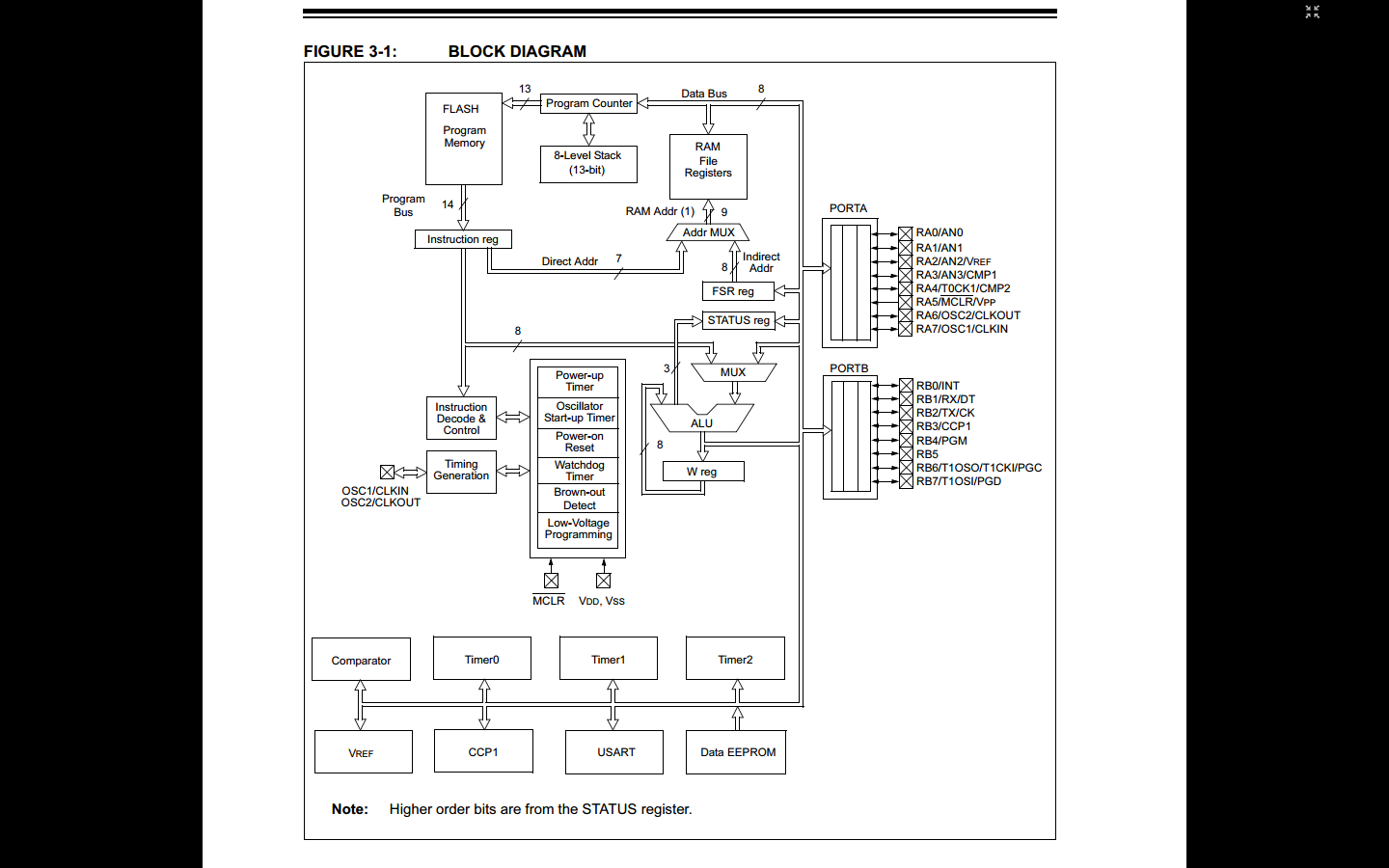
Figura - *Pinout* do PIC16F628A



Em sua estrutura interna, como vemos na Figura 3, contamos com a ULA – unidade lógica aritmética responsável pelas operações lógicas – diretamente ligada ao registrador W – registrador de trabalho ou acumulador. No canto superior esquerdo está a memória de programa, de onde sai um barramento de 14 bits. Ao centro está a memória de dados (RAM), que possui um barramento de 8 bits.

Do lado direito estão as portas com os pinos de entrada e saída. Na parte inferior estão os periféricos com que o PIC16F628A é capaz de trabalhar: EEPROM, *timers*, comparador de tensão, módulo CCP e a porta serial. Mais ao centro têm-se o registrador de status, na parte superior, o contador de linha de programa e a pilha de 8 níveis. Entre todos os periféricos a comunicação serial é feita por meio de um barramento de 8 vias.

Figura - Diagrama de blocos da estrutura interna do PIC16F628A. *Fonte:* [1].



No projeto em questão, é utilizada somente a comunicação serial assíncrona, que será esclarecida posteriormente, através do periférico USART, e o PORTA e o PORTB para controle do motor e acendimento dos LED’s.

Apesar do uso de comunicação serial, o que exigiria uma melhor precisão no *clock* do PIC, estaremos supondo que os dois microcontroladores do projeto estarão submetidos às mesmas condições de trabalho (na mesma sala), e, portanto, o oscilador interno (4,0 MHz) destes pode ser utilizado sem comprometimento da comunicação, sem necessidade de cristal/ressonador externo.

### *Universal Synchronous/Asynchronous Receiver/Transmitter* (USART)

O PIC16F628A possui um módulo de recepção e transmissão de dados por comunicação serial conhecido como USART. Os registradores TXSTA e RCSTA são responsáveis pela configuração do módulo USART nesse microcontrolador. O significado de cada *bit* desses registradores pode ser conferido na Figura 4 e na Figura 5.

Figura - *Bits* do registrador TXSTA (98h). *Fonte*: [1]

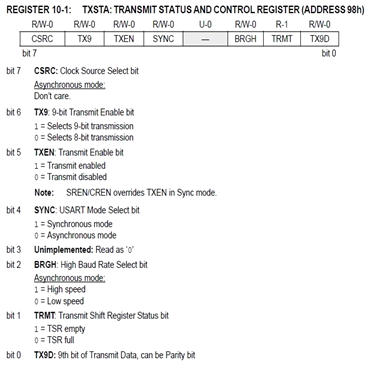
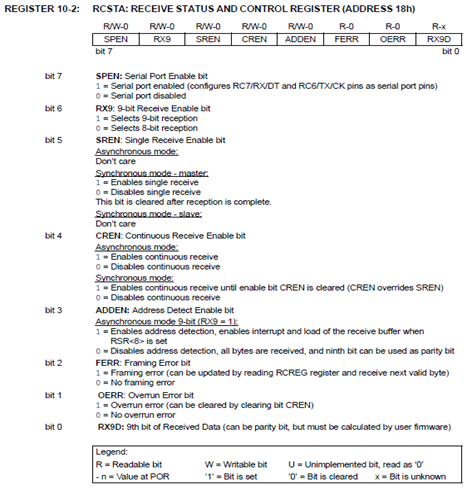


Figura - *Bits* do registrador RCSTA (18h). *Fonte*: [1]

******

Depois de o módulo ser configurado, basta carregar algum valor no registrador TXREG para que este seja enviado pelo pino TX do microcontrolador. Para a recepção, os dados são obtidos através do pino RX, e estes podem ser lidos pelo acesso ao registrador RCREG.

Além disso, faz-se necessária a configuração de *baud rate*, que se trata velocidade de transmissão de informações, em outras palavras, a quantidade de bits que podem ser transmitidos por segundo (bps). O *bit* BRGH (registrador TXSTA) define a velocidade de transmissão para duas diferentes faixas. O cálculo para comunicações assíncronas é dado abaixo:

Para o BRGH = 0 (baixa velocidade):

Para o BRGH =1 (alta velocidade):

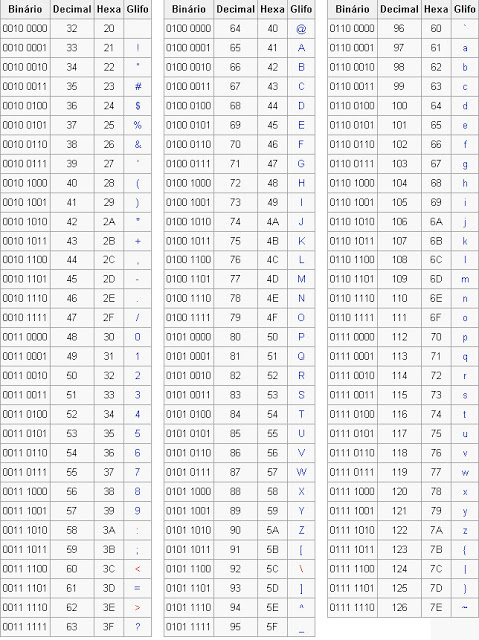
No projeto, utilizou-se a seguinte configuração no módulo USART:

* *Baud Rate*: ~ 9600 *bps* (BRGH <TXSTA> = 1, SPBRG = 25, = 4 MHz);
* 8 *bits* de dados (TX9 <TXSTA> = 0, RX9 <RCSTA> = 0);
* Modo assíncrono (SYNC <TXSTA> = 0);

## Tabela *ASCII*

Cada letra possui um código numérico correspondente, que pode ser representado tanto em hexadecimal (dois dígitos) como decimal (três dígitos) e binário (oito dígitos). Nos sistemas computacionais é comum a utilização da representação em hexadecimal e sua função é padronizar a forma como os computadores representam letras, números, acentos, sinais diversos e alguns códigos de controle.

Figura - Tabela ASCII



## Materiais utilizados e Custo Projeto.

Segue abaixo, na Tabela 1, um apanhado com todos os materiais, e seus respectivos custos, utilizados no Projeto.

Tabela - Materiais e Custo do Projeto.

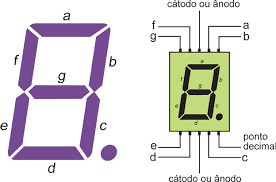
|  |  |  |  |
| --- | --- | --- | --- |
| Quantidade | Materiais | Preço Unitário | Preço |
| 1 | 1 Display 7 Segmentos | R$ 2,00 | R$ 2,00 |
| 8 | Resistores – 330 Ω | R$ 0,15 | R$ 1,20 |
| 3 | Resistores – 10 kΩ | R$ 0,15 | R$ 0,45 |
| 2 | Resistores – 560 Ω | R$ 0,15 | R$ 0,30 |
| 2 | LEDS difusos 5mm | R$ 0,50 | R$ 1,00 |
| 4 | PIC's 16F628A | R$ 14,00 | R$ 56,00 |
| 1 | Botão Switch | R$ 1,00 | R$ 1,00 |
| 1 | Regulador de Tensão L7805 | R$ 1,30 | R$ 1,30 |
| 1 | Barra de 40 pinos Macho | R$ 9,91 | R$ 9,91 |
| 1 | Módulo ULN | R$ 9,90 | R$ 9,90 |
| 2 | Capacitores Cerâmicos 100nF | R$ 0,12 | R$ 0,24 |
| 2 | Capacitores Eletrolíticos 470uF | R$ 0,25 | R$ 0,50 |
| 1 | Adaptador Jack para fonte | R$ 15,00 | R$ 15,00 |
| 2 | Soquetes para CI's 18 pinos | R$ 3,20 | R$ 6,40 |
| 1 | Motor de Passo | 0\* | R$ 0,00 |
| 1 | CD para base do Círculo (6 cm) | 0\* | R$ 0,00 |
|  |  | Preço Total | R$ 105,20 |

O motor de passo utilizado está com preço nulo por ter origem desconhecida, e não foi comprado para o uso. O CD foi de um membro da equipe e também não gerou custo adicional ao projeto.

## *Display* de 7 segmentos

O display de sete segmentos é um invólucro com sete LED’s em formato de segmento, organizados em um arranjo que possibilita a formação de números decimais e algumas letras utilizadas no código hexadecimal.

Figura – Ilustração de um display de 7 segmentos, com o seu pinout à direita.



A porta usada para acionar o *display* é o PORTA, entre os pinos RA0 e RA7 (pinos 17, 18, 1, 2, 3, 15 e 16, respectivamente), com exceção do RA5, que não pode ser usado para saída (isso é uma limitação do próprio microcontrolador). A escolha do tipo ânodo comum se tornou mais viável pelo fato de o RA4 (pino 3) ser dreno aberto, logo, não é capaz de “fornecer” corrente para um LED, apenas “absorver”.

Um *display* do tipo ânodo comum é aquele em que os terminais ânodo de todos os segmentos estão interligados internamente. Para cada um deles acender, este terminal comum foi conectado em 5V, enquanto que o cátodo do segmento, em série com um resistor adequado, será conectado ao GND.

O *datasheet* [2] do componente utilizado neste projeto (LEDS8012AUR1) recomenda uma corrente máxima de por segmento, e informa que a queda de tensão típica em cada segmento (ou ponto) é de . Assim, resistores de foram utilizados, com os quais estima-se uma corrente de acionamento em cada LED de cerca de:

No projeto, o *display* fornece a informação sobre qual letra está sendo apontada. Em que com *n* números, nos dois dígitos hexadecimais da primeira letra vê-se 1 e nos dois dígitos da última vê-se *n*. O ponto se refere à qual dígito hexadecimal (primeiro ou segundo) está sendo visto.

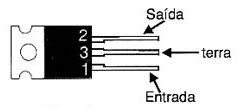
## Sobre o CI L7805CV

A série de circuitos integrados 78XX consiste em reguladores de tensão positiva com corrente de até 1 ampère de saída e que são apresentados em um invólucro TO-220.

O XX dessa série representa a tensão de saída do circuito, onde temos, por exemplo, o 7805 com uma tensão de saída de 5 volts e o 7812 com uma tensão de saída de 12 volts, etc.

A arquitetura do circuito, como mostrada na Figura 8, é de forma que um de seus pinos receba uma tensão a ser regulada, e o pino oposto seja de saída, onde se obtém a tensão prevista pelo CI. O pino do meio deve ser aterrado.

Figura – Pinout do L7805CV

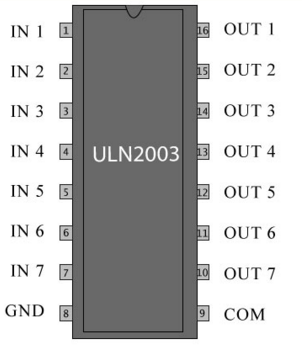


O regulador de tensão 7805 está sendo usado no projeto de forma que a tensão vinda da fonte de 12 volts (fonte de alimentação do projeto) possa ser regulada a uma tensão de 5 volts, a qual poderá alimentar os PIC’s presentes, já que os mesmos não suportariam a tensão original da fonte.

## Sobre o CI ULN2003

Para acionarmos um motor de passo precisamos de um hardware específico chamado driver. Um exemplo de driver é o ULN 2003, usado neste projeto. Este driver trata-se basicamente de um arranjo de transistores Darlington que podem controlar correntes de até 500 mA.

Figura – Pinout do ULN2003

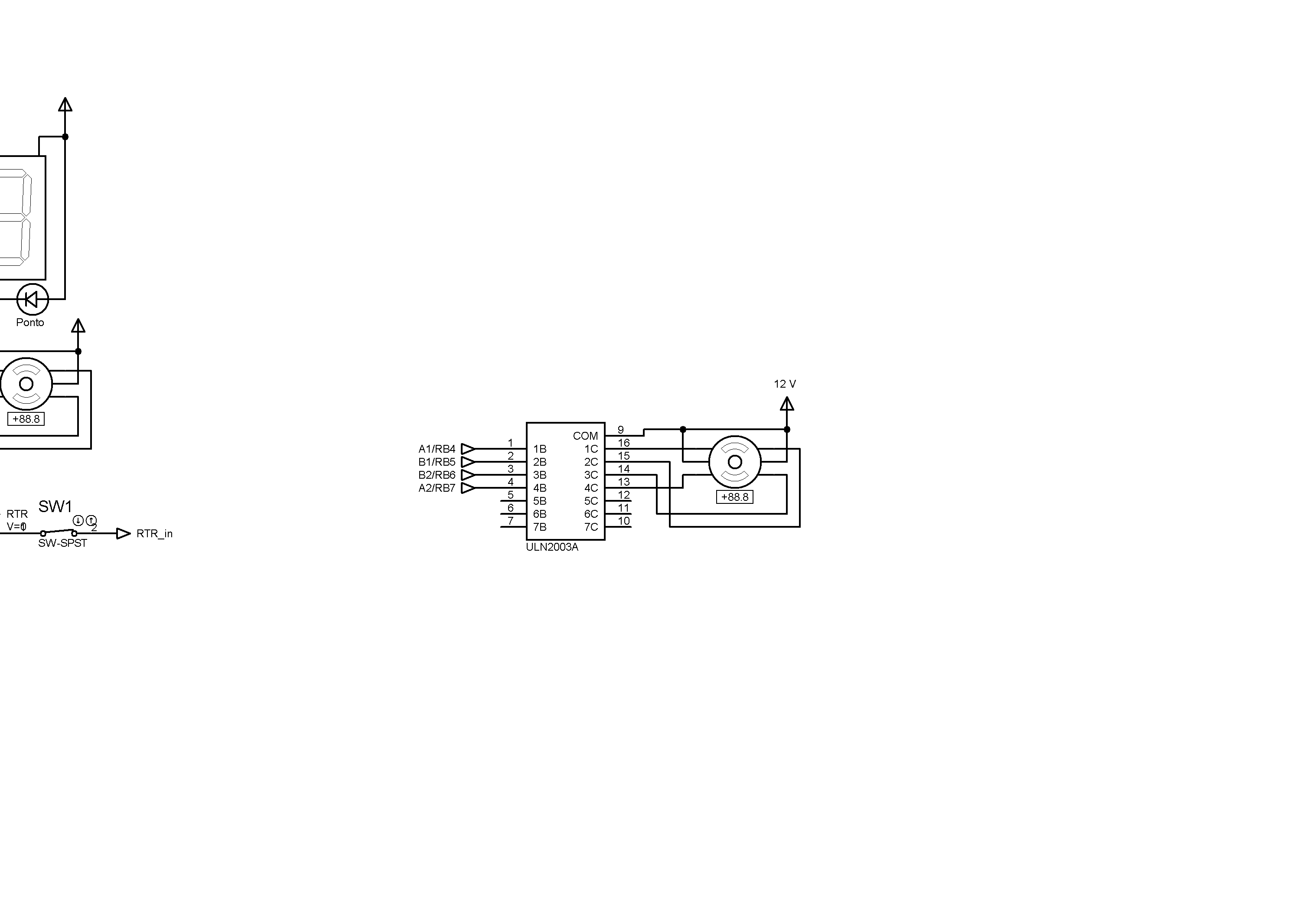


Como pode ser visto na Figura 9, este circuito integrado tem seus sete primeiros pinos como entrada, o oitavo pino sendo o Terra, o nono pino de alimentação comum à alimentação do motor e os sete últimos pinos como saída.

Cada saída possui diodos de roda-livre para acionamento de cargas indutivas, fazendo com que este driver seja ideal para acionamento do motor de passo unipolar, com ligação feita como no esquemático apresentado na Figura 10.

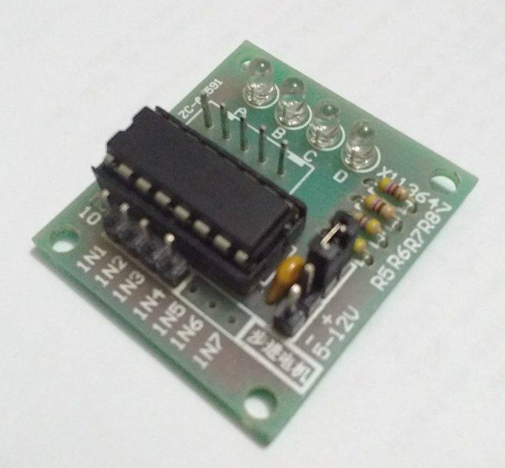
Figura – Esquemático de interfaceamento entre sinais de controle (A1, B1, B2, A2) e motor de passo por meio do circuito integrado ULN2003.

Imagem obtida por meio do *software* *Proteus 8 Professional.*



O circuito foi implementado numa placa, como mostrado na Figura 11, de forma a organizar a implementação do mesmo no projeto, pois as ligações foram feitas por meio de *jumpers* de acordo com o esquemático apresentado na Figura 10.

Figura – Placa para o ULN2003



## Motor de Passo

Motores de passo, devido ao seu funcionamento particular, podem ser controlados com um alto grau de precisão em malha aberta, ou seja, sem nenhuma necessidade de sensores de posição ou velocidade. Isso se deve ao fato de que eles se movimentam por meio de passos fixos de ângulo, de modo que para se alcançar uma determinada posição angular basta realizar uma quantidade adequada de passos.

Esses motores possuem bobinas alinhadas (chamadas de *fases*) que, quando acionadas, criam campos magnéticos que alinham o rotor ao seu eixo; quando há o acionamento sequencial das bobinas, é possível promover um deslocamento angular útil no rotor em qualquer sentido.

O motor utilizado para o movimento do ponteiro sobre o disco de hexadecimais trata-se de um motor de passo de duas fases, ambas com derivação central, o que o caracteriza como um motor *unipolar*, assim como pode ser conferido na Figura 12. As derivações centrais foram soldadas unidas, como mostrado na Figura 13, de modo que nesse terminal comum é fornecida a alimentação positiva da fonte externa.

Figura – Motor unipolar.

Os terminais conectados aos centros das duas bobinas estão demarcados na figura à esquerda por 1 e 2. *Fonte:* [2].

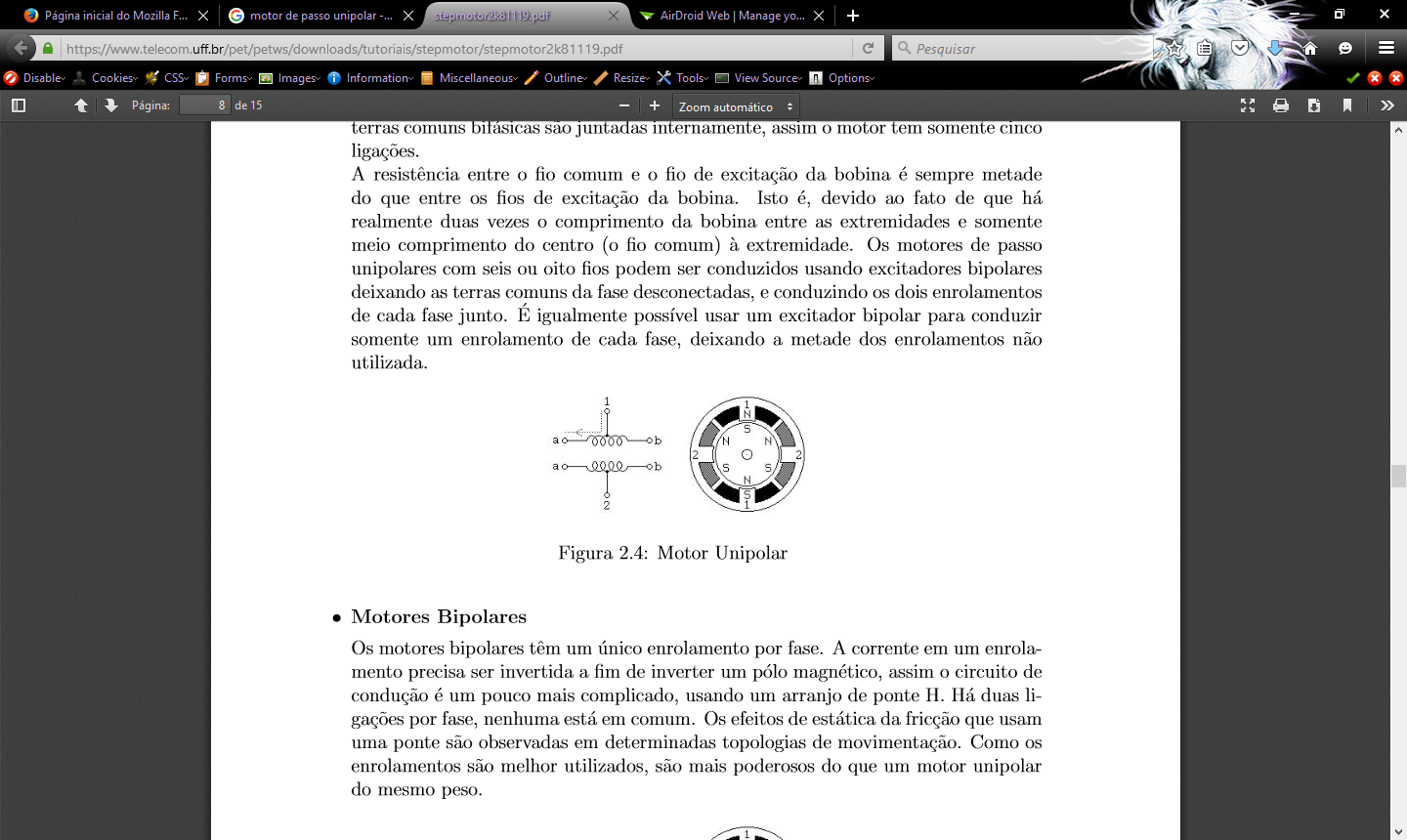
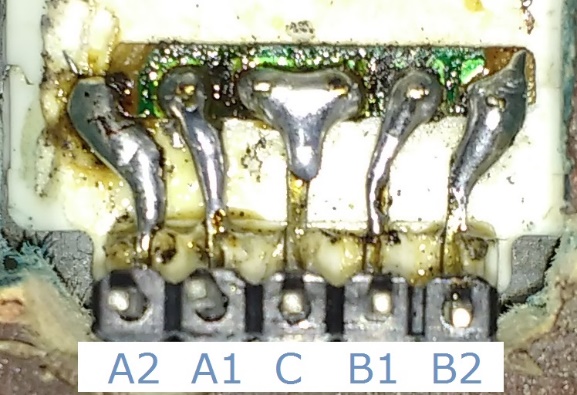


Figura – Terminais do motor de passo utilizado.

O terminal central é conectado às derivações centrais de cada fase. *Fonte:* o próprio autor.



### Acionamento do Motor

Cada uma das duas extremidades de ambas as fases é conectada a uma saída de um canal diferente do CI ULN2003 (Figura 9). Assim como mencionado no tópico anterior, quando um sinal alto for colocado na entrada de um canal do ULN2003, a saída desse canal é conectada ao terra, fazendo fluir uma corrente a partir do terminal comum do motor de passo em direção à extremidade da bobina que está conectado à essa mesma saída.

Apesar da procedência desconhecida do motor empregado e da dificuldade em encontrar especificações técnicas sobre este, alguns testes foram realizados a fim de determinar características essenciais quem permitam sua aplicação. Algumas delas se encontram na Tabela 2.

Tabela - Características do motor utilizado.

*Fonte:* o próprio autor.

|  |  |
| --- | --- |
| **Características Gerais** | |
| **Tipo de Motor** | Passo / Unipolar |
| **Número de fases** | 2 (duas) |
| **Número de terminais** | 5 (cinco) / um terminal comum a duas fases |
| **Número de passos por revolução** | 48 passos |
| **Regime de passos adotado** | Meio-passo (96 passos ao todo) |
| **Características Elétricas** | |
| **Tensão nominal** | 12 V |
| **Resistência elétrica média** | 160 Ω (Fase-Fase) / 80 Ω (Fase-Comum) |
| **Corrente média nominal por fase** | 150 mA (Fase-Comum) |

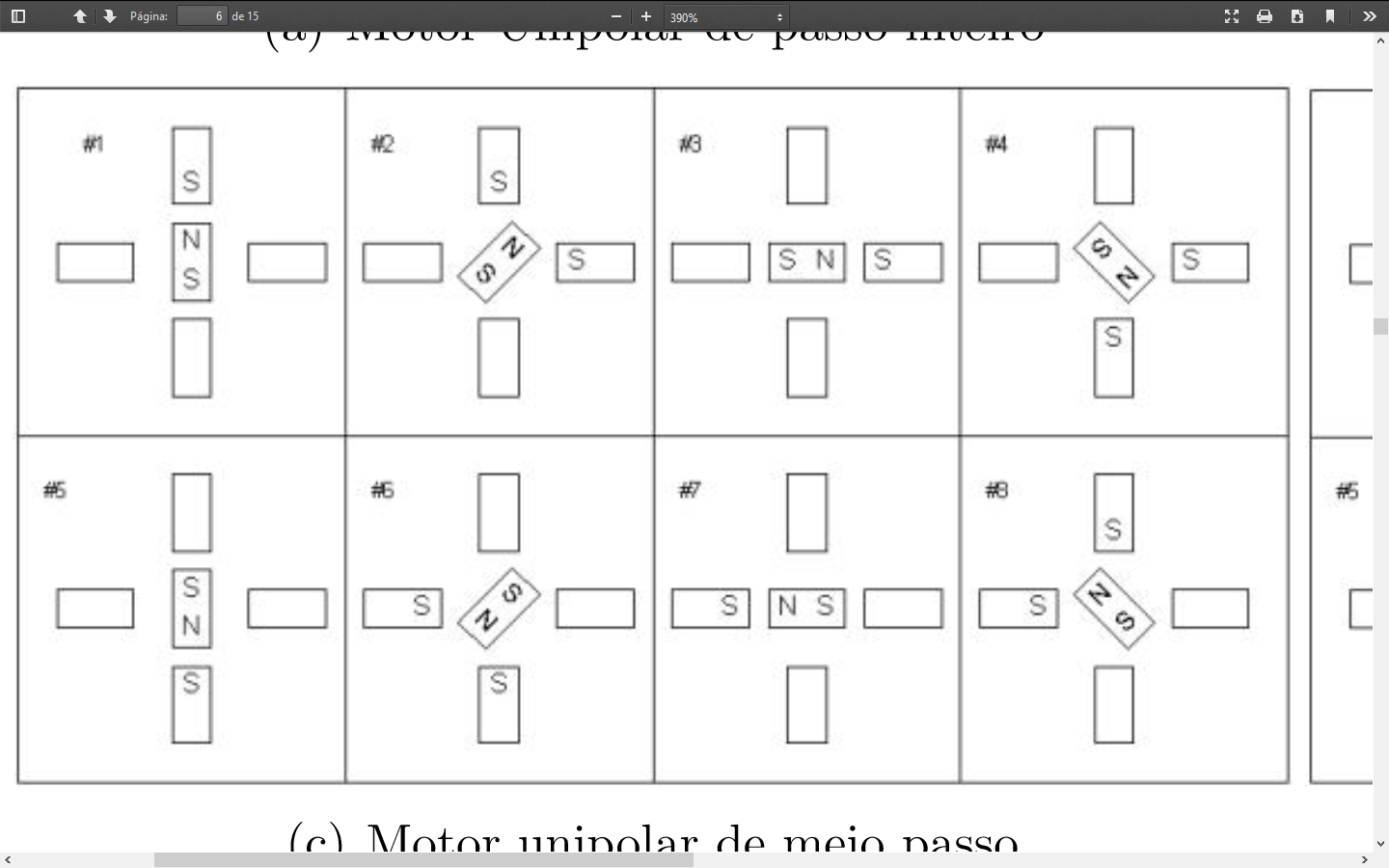
A fim de obter-se uma melhor resolução dos passos, o acionamento das bobinas se deu sob o regime de meio-passo. Dessa forma, a sequência de acionamento possui 8 diferentes estados, mostrados na Tabela 3 e simplificadamente ilustrados na Figura 14.

Tabela - Sequência de estados.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Número do estado | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Estado de acionamento  **A1**/**B1**/**B2**/**A2** | 0001 | 0011 | 0010 | 0110 | 0100 | 1100 | 1000 | 1001 |

Figura – Figura ilustrativa demonstrando o funcionamento do meio-passo em um motor unipolar.

*Fonte:* [2].



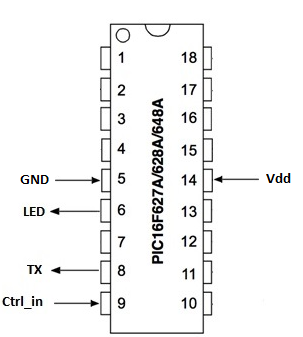
A partir desse ponto do relatório, cada “meio-passo” será referenciado apenas como “passo”, com o objetivo de tornar mais claro a explanação dos procedimentos.

## PIC de Dados

O PIC de Dados é o PIC no qual será gravado a palavra a qual será convertida pelo programa e decodificada. É comunicada pela serial com o PIC de Controle que é o responsável pela decodificação do dado.

### *Pinout*

Figura – Pinout do PIC de Dados

****

**Ctrl\_in** – Pino 9 do PIC de Dados (RB3) – Utilizado para 3 coisas em síntese: Pedido de novo dado, comando para ligar LED e comando para desligar LED (LED para indicar que chegou na parte alta e depois parte baixa).

**Tx** – Pino 8 (RB2) – Pino mais importante do PIC de DADOS, já que é o pino que de fato envia os dados pela serial para o PIC de Controle.

**LED**: Pino 6 (RB0) – Pino para LED de indicação de chegada em valor hexadecimal. Fica ligado 3 segundos, tempo que o ponteiro fica parado no número.

### Programação

**Configuração:**

Inicialmente foram feitas as configurações necessárias para o funcionamento correto do programa, ou seja, as configurações de entradas e saídas, de transmissão e recepção serial, de interrupções. O PORTA foi ajustado todo para entrada (42), com os RA2, RA1 e RA0 ajustados para digital. O PORTB foi ajustado com apenas o PORTB,2 (TX) e o PORTB,0 como saídas. Transmissão serial habilitada com baud-rate de 9600 (52), assíncrona, sem bit de paridade, ou seja, 8 bits, em alta (por isso o 9600).

Os caracteres a serem enviados são tomados a partir de uma tabela na rotina “DADOS”, como será visto adiante.

**Envio dos Dados:**

ENVIO\_DADOS:

Testa o Control\_in, bit que indica várias coisas que serão explicadas posteriormente. Nesse ponto, indica pedido de novo dado (75). Dado é entregue a partir da rotina “Dados”, e é enviado. A partir daí espera que o Ctrl\_in vá para zero (86), o que acontece quando o PIC de Controle recebe um dado.

Então espera que esse sinal vá para 1, quando termina a parte alta do código ASCII (89). Acende um LED, espera que esse sinal (Ctrl\_in) seja mandado para zero, depois de 3 segundos, e então apaga LED. Repete esse processo ao terminar parte baixa da palavra ASCII, ou seja, o segundo número hexadecimal (linhas 96 a 103). Ao terminar esses processos, recomeça o processo de enviar dados, novamente testando Ctrl\_in, como explicado anteriormente.

DADOS (29):

Utilizada na busca dos dados para o envio, pois é onde se encontra a tabela com os valores que serão enviados. Essa tabela é encontrada a partir da diretiva DT, a qual substitui os caracteres colocados nela em uma tabela de RETLW. Assim, ao somar a contagem com PCL, retorna em W o caractere ASCII que deve ser enviado. Ao terminar esse processo, ou seja, chamar todos os caracteres, fica preso num loop infinito e não envia mais nenhum dado.

### Código – PIC de Dados

Quadro - PIC de Controle

1

2 #include "P16F628A.inc"

3

4 ; CONFIG

5 ; \_\_config 0x3F3C

6 \_\_CONFIG \_FOSC\_INTOSCIO & \_WDTE\_OFF & \_PWRTE\_OFF & \_MCLRE\_ON & \_BOREN\_OFF & \_LVP\_OFF & \_CPD\_OFF & \_CP\_OFF

7

8

9 #DEFINE BANK0 BCF STATUS, RP0

10 #DEFINE BANK1 BSF STATUS, RP0

11

12 #DEFINE Ctrl\_in PORTB, 3

13 #DEFINE LED PORTB, 0

14

15 CBLOCK 0x70

16 Contagem

17 VAR1

18 VAR2

19 VAR3

20 ENDC

21

22 ORG 0x0000

23 GOTO INICIO

24

25 ORG 0x0004 ; Se houver alguma interrupção, programa retorna

26 RETFIE

27

28 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

29 DADOS:

30 MOVF Contagem, W

31 ADDWF PCL ; Soma com PCL para apontar para o endereço.

32 DT "Star Wars" ; DT substitui uma sequência de RETLW e funciona como tal

33 GOTO $ ; Quando terminar palavras, programa fica preso.

34 CLRF Contagem

35

36

37 INICIO:

38 BCF STATUS, RP1 ; Fixar bancos 0 e 1

39

40 BANK1

41 MOVLW 0xFF

42 MOVWF TRISA ; PORTA todo como entrada.

43 MOVLW B'11111010'

44 MOVWF TRISB ; Bits 2 e 0 do PORTB como saída, resto entrada.

45

46 BANK0

47 BSF LED

48 CALL DELAY\_START ; Espera um pouco para começar

49 BCF LED

50 BANK1

51

52 MOVLW .25 ; Define velocidade de 9600 Bauds/s.

53 MOVWF SPBRG

54

55 MOVLW B'00100100'

56 MOVWF TXSTA ; Habilita transmissão e seleciona alta

57

58 MOVLW B'10000000' ; Apenas desabilita resistores pull-up.

59 MOVWF OPTION\_REG

60

61 BANK0

62 MOVLW B'00000111'

63 MOVWF CMCON

64

65 MOVLW B'10000000'

66 MOVWF RCSTA ; Habilita porta serial

67

68 MOVLW B'11000000'

69 MOVWF INTCON ; Habilita interrupção geral e de periféricos

70

71 BCF LED ; Apaga led e começa.

72 CLRF Contagem

73

74 ENVIA\_DADO: ; Rotina enviar dado

75 BTFSS Ctrl\_in ; Ctrl Setado? Pode enviar dado?

76 GOTO $-1 ; Não, volta.

77

78 CALL DADOS ; Sim, envia dado.

79 MOVWF TXREG

80 BANK1

81 BTFSS TXSTA, TRMT

82 GOTO $-1

83 BANK0

84 INCF Contagem ; Incrementa contagem a cada chamada e envio.

85

86 BTFSC Ctrl\_in ; Dado foi recebido?

87 GOTO $-1 ; Não, volta a testar.

88

89 BTFSS Ctrl\_in ; Sim, Terminou a parte alta?

90 GOTO $-1 ; Não, retorna

91 BSF LED ; Sim, liga o led de aviso de parada.

92

93 BTFSC Ctrl\_in ; Recebeu aviso de 3s passados?

94 GOTO $-1 ; Não, volta.

95 BCF LED ; Sim, apaga led.

96

97 BTFSS Ctrl\_in ; Terminou parte baixa?

98 GOTO $-1 ; Não, retorna

99 BSF LED ; Sim, liga o led de aviso de parada.

100

101 BTFSC Ctrl\_in ; Recebeu aviso de 3s passados?

102 GOTO $-1 ; Não, volta.

103 BCF LED ; Sim, apaga led.

104

105 GOTO ENVIA\_DADO ; Volta ao inicio rotina de enviar dado

106

107 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

108

109 DELAY\_START

110 ;2999995 cycles

111 movlw 0x1A

112 movwf VAR1

113 movlw 0x8B

114 movwf VAR2

115 movlw 0x07

116 movwf VAR3

117 DELAY\_FLAG\_0

118 decfsz VAR1, f

119 goto $+2

120 decfsz VAR2, f

121 goto $+2

122 decfsz VAR3, f

123 goto DELAY\_FLAG\_0

124

125 ;1 cycle

126 nop

127

128 ;4 cycles (including call)

129 return

130

131 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

**PCB – Placa de Circuito Impresso – PIC de Dados**

Foi confeccionada uma placa para o PIC de Dados, nela temos um PIC16F628A,4 resistores (3 de 10kΩ e 1 de 560Ω), 2 LED’S, 4 capacitores (2 de 100nF e 2 de 470uF), 2 barras de pinos e um regulador de tensão.

Figura – PCB PIC de Dados.

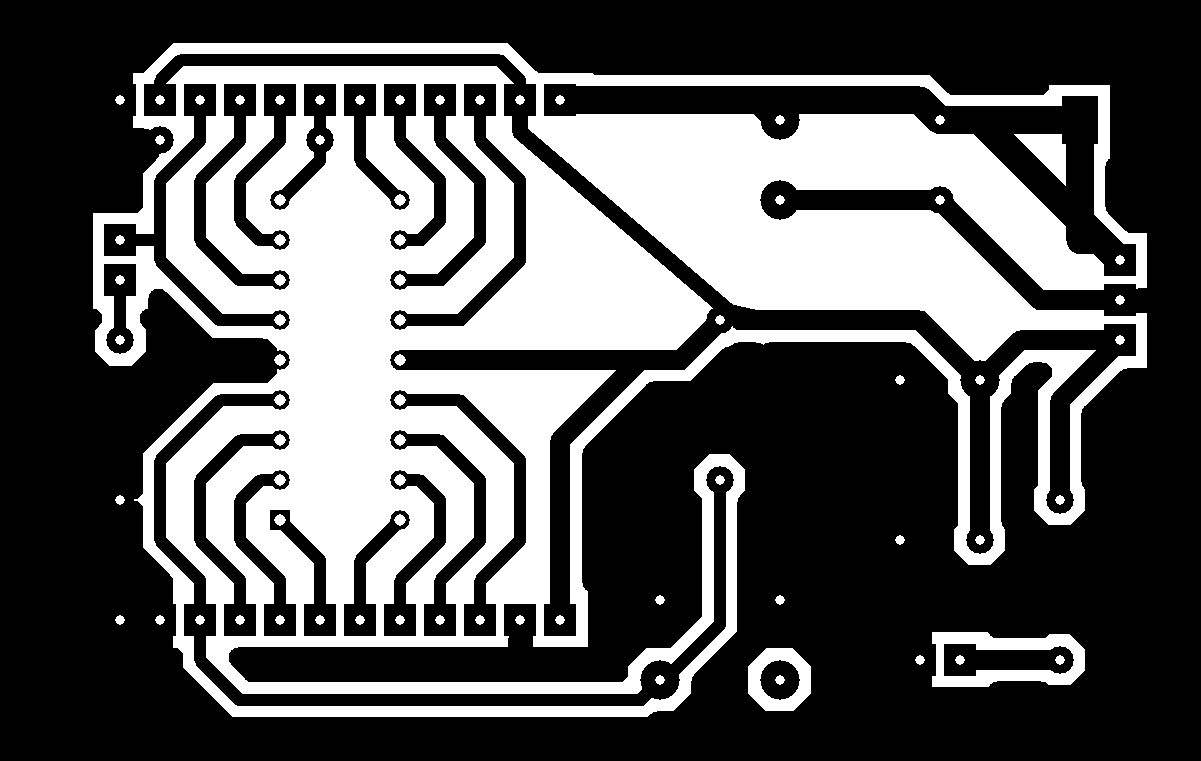


Figura – Placa simulada em 3D no Proteus.

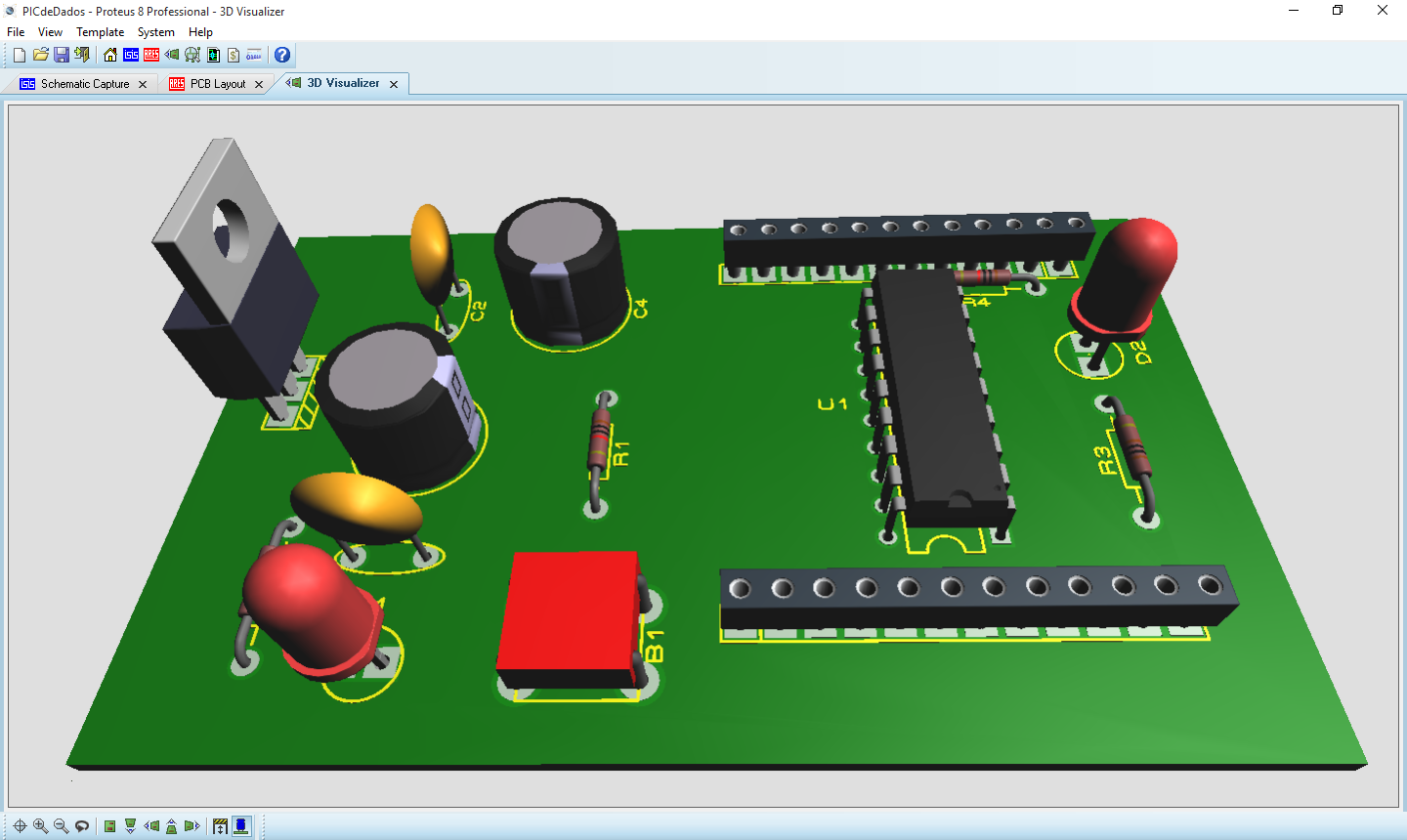
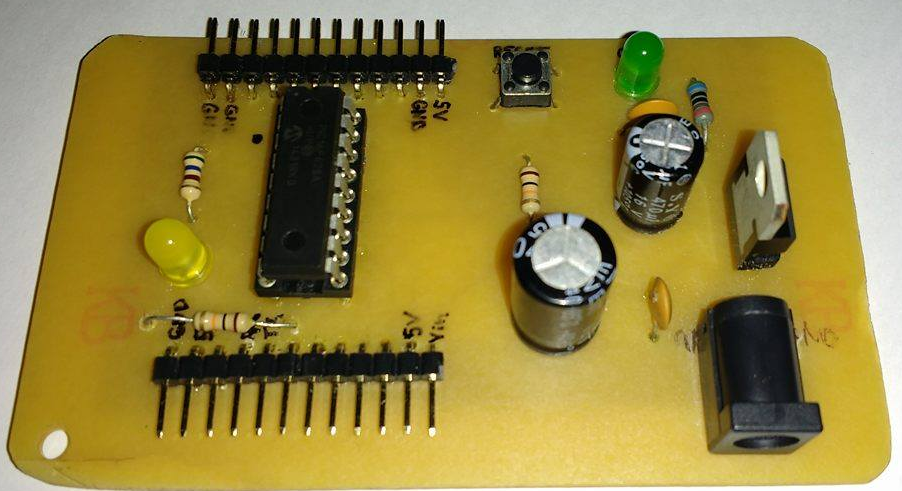


Figura – Placa PIC de Dados



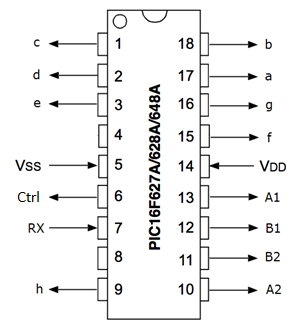
## PIC de Controle

O segundo microcontrolador utilizado neste projeto será aqui chamado PIC de Controle. Este está diretamente conectado ao CI ULN2008, e ambos, em conjunto, são os responsáveis pelo interfaceamento com o usuário através do controle do motor de passo.

### *Pinout*

A Figura 19 mostra o *pinout* de cada um dos pinos do PIC de Controle.

Figura - *Pinout* do PIC de Controle.



* **VDD (14) / VSS (5)**: Pinos reservados para a alimentação do PIC.
* **a (17) / b (18) / c (1) / d (2) / e (3) / f (15) / g (16) / h (9):** Saídas conectadas a cada um dos LED’s do *display* de 7 segmentos, incluindo o ponto decimal (*h*).
* **RX (7)**: Terminal de entrada de dados por comunicação serial. É conectado ao terminal TX do PIC de Dados.
* **Ctrl (6) –** Sinal de Controle: Terminal de saída que indica o estado de leitura da comunicação serial e realiza solicitações de novos dados ao PIC de Dados. Conectado ao terminal Ctrl\_in do PIC de Dados.
* **A2 (10) / B2 (11) / B1 (12) / A1 (13):** Sinais que controlam o estado de energização das bobinas do motor de passo. Estes terminais são conectados a entradas de canais do CI ULN2003 ().

### Constantes de programa

Ao todo, são utilizados no programa oito registradores de propósito geral (GPR), cada um com um significado próprio no desenvolvimento do projeto. Todos os endereços receberam “apelidos” definidos no bloco de constantes – CBLOCK –, linha 21, Quadro 2 (página 27):

* BYTE (0x70): Guarda o último valor recebido pela serial. Ocasionalmente é manipulado com inversão dos *nibbles*.
* POSITION (0x72): É frequentemente alterado para corresponder à posição instantânea do motor na roda de dígitos hexadecimais.
* TARGET (0x71): Valor da posição para o qual o motor irá convergir. A rotina de controle verifica a diferença entre TARGET e POSITION e a cada iteração, procura reduzi-la, até que POSITION se iguale a TARGET.
* DIF (0x73): Armazena a diferença entre os dois registradores anteriores. Eventualmente, o valor contido em DIF poderá ser um valor negativo, que estará escrito em complemento de 2. A rotina de controle se baseia no valor de DIF para mover o motor: dependendo do sinal, o sentido poderá ser horário ou anti-horário.
* BOBINA\_8 (0x74): É empregado para o correto sequenciamento do acionamento das bobinas do motor. Como o motor de passo trabalhando a meio-passo só terá 8 estados distintos de acionamento, esta variável só conterá valores de 0 a 7, sendo estes especificados anteriormente em Motor de passo.
* CONTAGEM (0x75): Guarda o valor da contagem de caracteres recebidos para ser mostrado no *display*.
* VAR1 (0x76), VAR2 (0x77) e VAR3 (0x78): São variáveis genéricas utilizadas para guardar valores temporários que estão sendo tratados ou contagem em rotinas de atraso.

Outras duas constantes definidas são PASSOS e PASSOS\_2, que correspondem, respectivamente, ao número total de passos e a metade desse valor.

### Programação

Toda a descrição dos parágrafos abaixo refere-se ao código do Quadro 2 da página 27. Em parênteses estão as linhas deste código as quais são descritas em cada parágrafo.

Inicialmente, o PIC de Controle é configurado por *software* para que todos os pinos utilizados no projeto sejam corretamente definidos como entrada ou saída, assim como mostra a Figura 19. A seguir, a comunicação serial é configurada para trabalhar em modo de recepção com uma velocidade de 9600 *bauds* por segundo, idêntica à do PIC de Dados, e é dado um estado inicial aos GPR’s e portas utilizados no programa. A interrupção de recepção serial é habilitada e só então o sinal Ctrl é colocado em nível alto, para solicitar ao PIC de Dados o primeiro dado (linhas 54 a 90).

Ao entrar na rotina principal de execução, o microcontrolador entra em um estado de espera, no qual aguarda a recepção de algum dado pelo terminal RX (linhas 95 a 97). A ocorrência desse evento aciona a interrupção de recepção serial, que salva o dado recebido em uma variável chamada BYTE. Dentro dessa rotina de interrupção, o sinal Ctrl é colocado em nível baixo (informando que o PIC está ocupado) e a recepção de dados via serial é desativada. (linhas 42 a 49).

Como a recepção é desativada, qualquer dado enviado para o PIC de Controle a esse ponto é ignorado. Entretanto, a correta configuração do PIC de Dados irá evitar que informações transmitidas sejam perdidas, uma vez que ele deverá constatar que o sinal Ctrl se encontra em nível baixo e aguardar novas ordens.

A próxima tarefa do microcontrolador é escrever no *display* o número de caracteres já recebidos (que na primeira execução dessa rotina é “1”) chamando a subrotina DISPLAY (linhas 99 e 100). Essa rotina simplesmente incrementa CONTAGEM e retorna em W o valor a ser escrito em PORTA. O ponto decimal do *display* é desligado (\_PONTO é setado – linha 102).

A subrotina TRATA é chamada para tomar primeiramente o *nibble* mais significativo de BYTE e, a partir de uma tabela, obter o valor que será gravado em TARGET (linha 103). Uma vez obtida a posição para o qual o motor deve ir, a rotina TRAJETO é chamada. Esta é composta por duas sub-rotinas principais, CALC e MOVIMENTO, descritas a seguir.

CALC

Esta sub-rotina (linha 211) é executada somente uma vez depois que TRAJETO é chamada. A partir da diferença entre as variáveis POSITION e TARGET, DIF é alterada para refletir o sentido e a quantidade de passos que o motor terá que executar para chegar à posição final.

A maneira ótima com que o algoritmo de cálculo foi implementado permite que DIF seja sempre um número em módulo menor ou igual à metade do número total de passos, pois, se porventura fosse maior, a contagem no sentido contrário certamente renderia um número menor de passos.

MOVIMENTO

Em conjunto com a sub-rotina interna MOVER (248 a 299), MOVIMENTO (linhas 191 a 205) é responsável por finalmente modificar sequencialmente o estado do conjunto dos quatro terminais de saída que controlam a energização das bobinas do motor de passo. Ela monitora o estado da variável DIF, e se mantem em um laço que a cada iteração incrementa ou decrementa unitariamente as variáveis DIF (até que seja igual a zero), POSITION (até que se iguale ao valor em TARGET) e BOBINA\_8 (que contem valores de 0 a 7 para cada estado de energização das bobinas do motor de passo). Entre cada laço é dado um *delay* (DELAY\_PASSO) para dar tempo de o motor energizar suas bobinas e realizar a rotação.

A partir de uma tabela (TABELA\_BOBINA), o valor de BOBINA\_8 é decodificado e gravado nos quatro terminais de saída. Dentro de MOVER, BOBINA\_8 e POSITION são continuamente testadas e modificadas para que se mantenham nos limites de 0 a 7 e de 0 a *n*-1, respectivamente (para *n* o número total de passos por revolução do motor definida na constante PASSOS). Um pequeno intervalo de tempo entre dois passos para estabilizar a corrente nas bobinas.

Conforme dito anteriormente, a sub-rotina MOVIMENTO se repete até que o motor esteja na posição que corresponde ao dígito hexadecimal mais significativo. Assim que o alcança, Ctrl é setado (106), dessa vez para ordenar o acendimento do LED de parada controlado pelo PIC de Dados. Um intervalo de tempo definido pela rotina DELAY\_FLAG é dado, e logo depois Ctrl volta a ser posto em nível baixo (108) para ordenar que o LED de parada do PIC de Dados se apague. Todo o processo anterior, desde o desligamento do ponto decimal do *display* (que dessa vez deverá ser apagado ao colocar \_PONTO = 0) até o desligamento do LED de parada do PIC de Dados é repetido para o *nibble* menos significativo.

Nesse ponto, o PIC de Controle já processou todos os dados que recebeu, então a recepção serial é reativada e o sinal Ctrl é novamente colocado em nível alto para informar que o sistema está pronto para receber novos comandos. Assim, o microcontrolador entra outra vez no estado de ociosidade, no qual aguarda uma nova transmissão serial pelo PIC de Dados.

#### Comentários gerais sobre o programa

A justificativa para que o LED de parada não tenha sido colocado em algum pino do PIC de Controle (o que é mais intuitivo, facilitaria a programação e permitiria que o sinal Ctrl fosse utilizado somente como um sinal RTR – “*Ready to Receive*”) é que não havia mais pinos disponíveis nesse PIC: dos 18, 2 deles eram para alimentação, 8 deles estavam dedicados ao *display*, 4 deles ao controle do motor de passo, 2 deles à comunicação serial (RX e TX), 1 deles dedicados ao próprio sinal Ctrl e o último (RA5) só pode ser utilizado como entrada.

O fato de a comunicação serial estar habilitada impediu o uso do pino TX (RB2) como saída de sinal digital. Embora fosse plausível cogitar a possibilidade de se transmitir via serial a solicitação de um novo dado ou para ligar/desligar o LED de parada, testes realizados mostraram falha nesse procedimento ou dificuldade na sincronização do tempo em que o LED fica aceso com o tempo em que o motor fica parado no setor.

O uso do Ctrl tanto para solicitação de novos dados como para controle do LED de parada se mostrou, portanto, mais eficaz nos testes realizados.

### Código – PIC de Controle

1 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

2 ; Projeto MICROS

3 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

4 #INCLUDE <P16F628A.INC>

5 \_\_CONFIG \_FOSC\_INTOSCIO&\_WDTE\_OFF&\_PWRTE\_OFF&\_MCLRE\_OFF&\_BOREN\_OFF&\_LVP\_OFF&\_CPD\_OFF&\_CP\_OFF

6 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

7 ; DEFINICOES

8 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

9 #DEFINE BANK0 BCF STATUS, RP0

10 #DEFINE BANK1 BSF STATUS, RP0

11

12 #DEFINE Ctrl PORTB, 0 ; "Sinal de controle"

13 #DEFINE \_PONTO PORTB, 3 ; Ponto decimal que indica qual hexadecimal está sendo mostrado

14

15 PASSOS EQU .96 ; Numero total de passos (contando meio-passo)

16 PASSOS\_2 EQU .48

17

18 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

19 ; REGISTRADORES DE USO GERAL

20 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

21 CBLOCK 0x70

22 BYTE ; Guarda o valor recebido pela serial

23 TARGET ; Indica a posicao para a qual o motor deve ir\*

24 POSITION ; Indica a posicao atual do motor\*

25 DIF ; Usado para iteracoes

26 BOBINA\_8 ; 0-7 : Indica, a partir da tabela 1, o estado de acionamento das bobinas

27 CONTAGEM ; Valor a ser mostrado no display (indica o ï¿½ndice do caractere)

28 VAR1

29 VAR2

30 VAR3

31 ENDC

32

33 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

34 ; START

35 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

36 ORG 0X00

37

38 GOTO MAIN

39 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

40 ; INTERRUPCAO

41 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

42 ORG 0X04

43

44 BANK0

45 MOVF RCREG, W

46 MOVWF BYTE

47 BCF Ctrl ; Desativa o pedido de dados a linha

48 BCF RCSTA, CREN ; Desabilita a interrupcao serial enquanto nao processar o valor atual

49 RETFIE

50

51 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

52 ; MAIN

53 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

54 MAIN:

55 BCF STATUS, RP1 ; Fixar em bancos 0 e 1

56

57 BANK1

58 CLRF TRISA ; Pinos do PORTA sao todos saida (exceto RA5)

59 MOVLW B'00000110'

60 MOVWF TRISB ;

61

62 MOVLW .25

63 MOVWF SPBRG ; BaudRate em 9600

64 BSF PIE1, RCIE ; Habilitar interrupcao de Recepcao Serial

65 MOVLW B'00100100'

66 MOVWF TXSTA

67

68 BANK0

69 MOVLW B'10010000'

70 MOVWF RCSTA ; Habilita comunicacao serial e recepcao continua.

71

72 MOVLW B'00000111'

73 MOVWF CMCON ; Desativar modulo comparador para usar RA0:RA3 como I/O

74

75 MOVLW B'10011111'

76 MOVWF PORTA ; Coloca "-" no display

77 MOVLW B'00110000'

78 CLRF CONTAGEM

79

80 MOVLW .1

81 MOVWF BOBINA\_8

82 CALL TABELA\_BOBINA

83 IORLW B'00001000'

84 MOVWF PORTB

85 CLRF POSITION

86

87 MOVLW B'11000000' ; Habilitar interrupcao Geral e de Perifericos

88 MOVWF INTCON

89

90 BSF Ctrl ; Solicita primeiro dado

91

92 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

93 ; ROTINA PRINCIPAL

94 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

95 LOOP:

96 BTFSC Ctrl ; Esse teste so se torna verdadeiro se recepcao serial gerar uma interrupcao

97 GOTO LOOP

98 ;---------------------------------------

99 CALL DISPLAY

100 MOVWF PORTA ; Escreve no display a contagem

101 ;---------------------------------------

102 BSF \_PONTO

103 CALL TRATA ; Target e' definido

104 CALL TRAJETO

105

106 BSF Ctrl

107 CALL DELAY\_FLAG

108 BCF Ctrl

109 ;---------------------------------------

110 BCF \_PONTO

111 CALL TRATA ; Move agora para a parte baixa

112 CALL TRAJETO

113

114 BSF Ctrl

115 CALL DELAY\_FLAG

116 BCF Ctrl

117 ;---------------------------------------

118 GOTO $+1

119 GOTO $+1 ; Pequeno delay para evitar erros de leitura pelo PIC de Dados

120

121 BSF RCSTA, CREN ; Reabilita a recepcao serial

122 BSF Ctrl ; Solicita novo dado a linha

123 GOTO LOOP

124

125 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

126 ; DISPLAY

127 ; Gerencia contagem (a partir de 1) de caracteres recebidos pela serial

128 ; Retorna valor a ser escrito em PORTA / pinout = fg\_edcba

129 ; Deve ser usado com display de sete segmentos do tipo ânodo comum

130 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

131 DISPLAY:

132 INCF CONTAGEM

133 MOVF CONTAGEM, W

134 ADDWF PCL

135 RETLW B'01000000' ; 0

136 RETLW B'11011001' ; 1

137 RETLW B'10000100' ; 2

138 RETLW B'10010000' ; 3

139 RETLW B'00011001' ; 4

140 RETLW B'00010010' ; 5

141 RETLW B'00000010' ; 6

142 RETLW B'01011000' ; 7

143 RETLW B'00000000' ; 8

144 RETLW B'00010000' ; 9

145 RETLW B'00001000' ; A

146 RETLW B'00000011' ; B

147 RETLW B'01000110' ; C

148 RETLW B'10000001' ; D

149 RETLW B'00000110' ; E

150 CLRF CONTAGEM

151 RETLW B'00001110' ; F

152

153 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

154 ; TRATAMENTO DE BYTE

155 ; Gerencia a ordem de leitura dos nibbles

156 ; Atualiza valor de TARGET

157 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

158 TRATA:

159 SWAPF BYTE

160 MOVLW B'00001111'

161 ANDWF BYTE, W

162 CALL TABELA\_POSICIONA

163 MOVWF TARGET

164 RETURN

165 TABELA\_POSICIONA:

166 ADDWF PCL

167 RETLW .3 ; 0

168 RETLW .9 ; 1

169 RETLW .15 ; 2

170 RETLW .21 ; 3

171 RETLW .27 ; 4

172 RETLW .33 ; 5

173 RETLW .39 ; 6

174 RETLW .45 ; 7

175 RETLW .51 ; 8

176 RETLW .57 ; 9

177 RETLW .63 ; A

178 RETLW .69 ; B

179 RETLW .75 ; C

180 RETLW .81 ; D

181 RETLW .87 ; E

182 RETLW .93 ; F

183

184 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

185 ; TRAJETO

186 ; Quando chamada, esta funcao faz com que POSITION (posicao do motor) se iguale a TARGET

187 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

188 TRAJETO:

189 CALL CALC ; Calcula trajetoria

190 ; Retorna em DIF a maneira otima de chegar a posicao final

191 MOVIMENTO:

192 CALL MOVER

193 ; { Modifica em PORTB somente os 4 bits que controlam o motor

194 MOVWF VAR1

195 MOVF PORTB, W

196 ANDLW B'00001111'

197 IORWF VAR1, W

198 MOVWF PORTB

199 ; }

200 CALL DELAY\_PASSO ; Delay adicional entre passos

201

202 MOVF DIF

203 BTFSS STATUS, Z ; DIF = 0?

204 GOTO MOVIMENTO ; Ainda nao...

205 RETURN ; Sim

206

207 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

208 ; CALCULA

209 ; Retorna em DIF a maneira otima de chegar a posicao final

210 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

211 CALC:

212 MOVF POSITION, W

213 SUBWF TARGET, W

214 MOVWF VAR1 ; VAR1 = TARGET - POSITION

215

216 BTFSS VAR1, 7 ; VAR1 e' negativo?

217 GOTO $+3

218 COMF VAR1, W

219 ADDLW .1 ; W = - VAR1

220

221 SUBLW PASSOS\_2 ; W = PASSOS/2 - |VAR1|

222

223 BTFSC STATUS, C ; Deu borrow?

224 GOTO NORMAL ; Não

225 ALTERADO: ; Transicao F <-- 0 ou F --> 0

226 MOVF VAR1, W

227 BTFSS VAR1, 7

228 GOTO $+3

229 ADDLW PASSOS

230 GOTO $+2

231 ADDLW -PASSOS

232 MOVWF DIF

233 RETURN

234 NORMAL:

235 MOVF VAR1, W

236 MOVWF DIF

237 RETURN

238

239 ; DIF guarda a quantidade de passos que devera ser dada.

240 ; Caso seja negativo, estara em complemento 2

241

242 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

243 ; MOVER

244 ; Move um passo na direcao determinada por DIF

245 ; Retorna em W o estado de energizacao das bobinas

246 ; Atualiza POSITION e DIF

247 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

248 MOVER:

249 MOVF DIF, F

250 BTFSC STATUS, Z ; DIF é zero?

251 GOTO TABELA\_BOBINA ; Sim

252

253 BTFSC DIF, 7 ; DIF é negativo?

254 GOTO CELERA\_ESQUERDA ; Sim

255

256 CELERA\_DIREITA:

257 DECF DIF

258

259 INCF POSITION

260 MOVLW PASSOS

261 XORWF POSITION, W

262 BTFSC STATUS, Z ; POSITION = PASSOS?

263 CLRF POSITION ; Sim, Entao Zera POSITION!

264

265 INCF BOBINA\_8

266 MOVLW .8

267 XORWF BOBINA\_8, W

268 BTFSC STATUS, Z ; BOBINA\_8 = 8?

269 CLRF BOBINA\_8 ; Sim, Entao Zera BOBINA\_8

270 GOTO TABELA\_BOBINA

271

272 CELERA\_ESQUERDA:

273 INCF DIF

274

275 MOVF POSITION

276 BTFSS STATUS, Z ; POSITION = 0 ?

277 GOTO $+3 ; Nao. So decrementa

278 MOVLW PASSOS ; Sim. Reinicia POSITION e decrementa

279 MOVWF POSITION

280 DECF POSITION ; de 0 a 2\*n-1

281

282 MOVF BOBINA\_8

283 BTFSS STATUS, Z ; BOBINA\_8 = 0 ?

284 GOTO $+3 ; Nao. So decrementa

285 MOVLW .8 ; Sim. Reinicia BOBINA\_8 e decrementa

286 MOVWF BOBINA\_8

287 DECF BOBINA\_8 ; de 0 a 7

288

289 TABELA\_BOBINA:

290 MOVF BOBINA\_8, W

291 ADDWF PCL

292 RETLW B'00010000' ; BOBINA\_8 = 0

293 RETLW B'00110000' ; BOBINA\_8 = 1

294 RETLW B'00100000' ; BOBINA\_8 = 2

295 RETLW B'01100000' ; BOBINA\_8 = 3

296 RETLW B'01000000' ; BOBINA\_8 = 4

297 RETLW B'11000000' ; BOBINA\_8 = 5

298 RETLW B'10000000' ; BOBINA\_8 = 6

299 RETLW B'10010000' ; BOBINA\_8 = 7

300

301 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

302 ; Rotinas de DELAY

303 ; DELAY\_FLAG é o tempo de espera adicional quando o motor chega a uma posição determinada

304 ; DELAT\_PASSO é o tempo adicional de espera entre os passos para energizar as bobinas

305 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

306 DELAY\_FLAG:

307 ;2999995 cycles

308 movlw 0x1A

309 movwf VAR1

310 movlw 0x8B

311 movwf VAR2

312 movlw 0x07

313 movwf VAR3

314 DELAY\_FLAG\_0

315 decfsz VAR1, f

316 goto $+2

317 decfsz VAR2, f

318 goto $+2

319 decfsz VAR3, f

320 goto DELAY\_FLAG\_0

321

322 ;1 cycle

323 nop

324

325 ;4 cycles (including call)

326 return

327 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

328 DELAY\_PASSO:

329 ;4993 cycles

330 movlw 0xE6

331 movwf VAR1

332 movlw 0x04

333 movwf VAR2

334 DELAY\_PASSO\_0

335 decfsz VAR1, f

336 goto $+2

337 decfsz VAR2, f

338 goto DELAY\_PASSO\_0

339

340 ;3 cycles

341 goto $+1

342 nop

343

344 ;4 cycles (including call)

345 return

346 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

347

348 END

Quadro - PIC de Controle

85 CLRF POSITION

86

87 MOVLW B'11000000' ; Habilitar interrupcao Geral e de Perifericos

88 MOVWF INTCON

89

90 BSF Ctrl ; Solicita primeiro dado

91

92 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

93 ; ROTINA PRINCIPAL

94 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

95 LOOP:

96 BTFSC Ctrl ; Esse teste so se torna verdadeiro se recepcao serial gerar uma interrupcao

97 GOTO LOOP

98 ;---------------------------------------

99 CALL DISPLAY

100 MOVWF PORTA ; Escreve no display a contagem

101 ;---------------------------------------

102 BSF \_PONTO

103 CALL TRATA ; Target e' definido

104 CALL TRAJETO

105

106 BSF Ctrl

107 CALL DELAY\_FLAG

108 BCF Ctrl

109 ;---------------------------------------

110 BCF \_PONTO

111 CALL TRATA ; Move agora para a parte baixa

112 CALL TRAJETO

113

114 BSF Ctrl

115 CALL DELAY\_FLAG

116 BCF Ctrl

117 ;---------------------------------------

118 GOTO $+1

119 GOTO $+1 ; Pequeno delay para evitar erros de leitura pelo PIC de Dados

120

121 BSF RCSTA, CREN ; Reabilita a recepcao serial

122 BSF Ctrl ; Solicita novo dado a linha

123 GOTO LOOP

124

125 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

126 ; DISPLAY

127 ; Gerencia contagem (a partir de 1) de caracteres recebidos pela serial

128 ; Retorna valor a ser escrito em PORTA / pinout = fg\_edcba

129 ; Deve ser usado com display de sete segmentos do tipo ânodo comum

130 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

131 DISPLAY:

132 INCF CONTAGEM

133 MOVF CONTAGEM, W

134 ADDWF PCL

135 RETLW B'01000000' ; 0

136 RETLW B'11011001' ; 1

137 RETLW B'10000100' ; 2

138 RETLW B'10010000' ; 3

139 RETLW B'00011001' ; 4

140 RETLW B'00010010' ; 5

141 RETLW B'00000010' ; 6

142 RETLW B'01011000' ; 7

143 RETLW B'00000000' ; 8

144 RETLW B'00010000' ; 9

145 RETLW B'00001000' ; A

146 RETLW B'00000011' ; B

147 RETLW B'01000110' ; C

148 RETLW B'10000001' ; D

149 RETLW B'00000110' ; E

150 CLRF CONTAGEM

151 RETLW B'00001110' ; F

152

153 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

154 ; TRATAMENTO DE BYTE

155 ; Gerencia a ordem de leitura dos nibbles

156 ; Atualiza valor de TARGET

157 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

158 TRATA:

159 SWAPF BYTE

160 MOVLW B'00001111'

161 ANDWF BYTE, W

162 CALL TABELA\_POSICIONA

163 MOVWF TARGET

164 RETURN

165 TABELA\_POSICIONA:

166 ADDWF PCL

167 RETLW .3 ; 0

168 RETLW .9 ; 1

169 RETLW .15 ; 2

170 RETLW .21 ; 3

171 RETLW .27 ; 4

172 RETLW .33 ; 5

173 RETLW .39 ; 6

174 RETLW .45 ; 7

175 RETLW .51 ; 8

176 RETLW .57 ; 9

177 RETLW .63 ; A

178 RETLW .69 ; B

179 RETLW .75 ; C

180 RETLW .81 ; D

181 RETLW .87 ; E

182 RETLW .93 ; F

183

184 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

185 ; TRAJETO

186 ; Quando chamada, esta funcao faz com que POSITION (posicao do motor) se iguale a TARGET

187 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

188 TRAJETO:

189 CALL CALC ; Calcula trajetoria

190 ; Retorna em DIF a maneira otima de chegar a posicao final

191 MOVIMENTO:

192 CALL MOVER

193 ; { Modifica em PORTB somente os 4 bits que controlam o motor

194 MOVWF VAR1

195 MOVF PORTB, W

196 ANDLW B'00001111'

197 IORWF VAR1, W

198 MOVWF PORTB

199 ; }

200 CALL DELAY\_PASSO ; Delay adicional entre passos

201

202 MOVF DIF

203 BTFSS STATUS, Z ; DIF = 0?

204 GOTO MOVIMENTO ; Ainda nao...

205 RETURN ; Sim

206

207 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

208 ; CALCULA

209 ; Retorna em DIF a maneira otima de chegar a posicao final

210 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

211 CALC:

212 MOVF POSITION, W

213 SUBWF TARGET, W

214 MOVWF VAR1 ; VAR1 = TARGET - POSITION

215

216 BTFSS VAR1, 7 ; VAR1 e' negativo?

217 GOTO $+3

218 COMF VAR1, W

219 ADDLW .1 ; W = - VAR1

220

221 SUBLW PASSOS\_2 ; W = PASSOS/2 - |VAR1|

222

223 BTFSC STATUS, C ; Deu borrow?

224 GOTO NORMAL ; Não

225 ALTERADO: ; Transicao F <-- 0 ou F --> 0

226 MOVF VAR1, W

227 BTFSS VAR1, 7

228 GOTO $+3

229 ADDLW PASSOS

230 GOTO $+2

231 ADDLW -PASSOS

232 MOVWF DIF

233 RETURN

234 NORMAL:

235 MOVF VAR1, W

236 MOVWF DIF

237 RETURN

238

239 ; DIF guarda a quantidade de passos que devera ser dada.

240 ; Caso seja negativo, estara em complemento 2

241

242 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

243 ; MOVER

244 ; Move um passo na direcao determinada por DIF

245 ; Retorna em W o estado de energizacao das bobinas

246 ; Atualiza POSITION e DIF

247 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

248 MOVER:

249 MOVF DIF, F

250 BTFSC STATUS, Z ; DIF é zero?

251 GOTO TABELA\_BOBINA ; Sim

252

253 BTFSC DIF, 7 ; DIF é negativo?

254 GOTO CELERA\_ESQUERDA ; Sim

255

256 CELERA\_DIREITA:

257 DECF DIF

258

259 INCF POSITION

260 MOVLW PASSOS

261 XORWF POSITION, W

262 BTFSC STATUS, Z ; POSITION = PASSOS?

263 CLRF POSITION ; Sim, Entao Zera POSITION!

264

265 INCF BOBINA\_8

266 MOVLW .8

267 XORWF BOBINA\_8, W

268 BTFSC STATUS, Z ; BOBINA\_8 = 8?

269 CLRF BOBINA\_8 ; Sim, Entao Zera BOBINA\_8

270 GOTO TABELA\_BOBINA

271

272 CELERA\_ESQUERDA:

273 INCF DIF

274

275 MOVF POSITION

276 BTFSS STATUS, Z ; POSITION = 0 ?

277 GOTO $+3 ; Nao. So decrementa

278 MOVLW PASSOS ; Sim. Reinicia POSITION e decrementa

279 MOVWF POSITION

280 DECF POSITION ; de 0 a 2\*n-1

281

282 MOVF BOBINA\_8

283 BTFSS STATUS, Z ; BOBINA\_8 = 0 ?

284 GOTO $+3 ; Nao. So decrementa

285 MOVLW .8 ; Sim. Reinicia BOBINA\_8 e decrementa

286 MOVWF BOBINA\_8

287 DECF BOBINA\_8 ; de 0 a 7

288

289 TABELA\_BOBINA:

290 MOVF BOBINA\_8, W

291 ADDWF PCL

292 RETLW B'00010000' ; BOBINA\_8 = 0

293 RETLW B'00110000' ; BOBINA\_8 = 1

294 RETLW B'00100000' ; BOBINA\_8 = 2

295 RETLW B'01100000' ; BOBINA\_8 = 3

296 RETLW B'01000000' ; BOBINA\_8 = 4

297 RETLW B'11000000' ; BOBINA\_8 = 5

298 RETLW B'10000000' ; BOBINA\_8 = 6

299 RETLW B'10010000' ; BOBINA\_8 = 7

300

301 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

302 ; Rotinas de DELAY

303 ; DELAY\_FLAG é o tempo de espera adicional quando o motor chega a uma posição determinada

304 ; DELAT\_PASSO é o tempo adicional de espera entre os passos para energizar as bobinas

305 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

306 DELAY\_FLAG:

307 ;2999995 cycles

308 movlw 0x1A

309 movwf VAR1

310 movlw 0x8B

311 movwf VAR2

312 movlw 0x07

313 movwf VAR3

314 DELAY\_FLAG\_0

315 decfsz VAR1, f

316 goto $+2

317 decfsz VAR2, f

318 goto $+2

319 decfsz VAR3, f

320 goto DELAY\_FLAG\_0

321

322 ;1 cycle

323 nop

324

325 ;4 cycles (including call)

326 return

327 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

328 DELAY\_PASSO:

329 ;4993 cycles

330 movlw 0xE6

331 movwf VAR1

332 movlw 0x04

333 movwf VAR2

334 DELAY\_PASSO\_0

335 decfsz VAR1, f

336 goto $+2

337 decfsz VAR2, f

338 goto DELAY\_PASSO\_0

339

340 ;3 cycles

341 goto $+1

342 nop

343

344 ;4 cycles (including call)

345 return

346 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

347

348 END

183

184 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

185 ; TRAJETO

186 ; Quando chamada, esta funcao faz com que POSITION (posicao do motor) se iguale a TARGET

187 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

188 TRAJETO:

189 CALL CALC ; Calcula trajetoria

190 ; Retorna em DIF a maneira otima de chegar a posicao final

191 MOVIMENTO:

192 CALL MOVER

193 ; { Modifica em PORTB somente os 4 bits que controlam o motor

194 MOVWF VAR1

195 MOVF PORTB, W

196 ANDLW B'00001111'

197 IORWF VAR1, W

198 MOVWF PORTB

199 ; }

200 CALL DELAY\_PASSO ; Delay adicional entre passos

201

202 MOVF DIF

203 BTFSS STATUS, Z ; DIF = 0?

204 GOTO MOVIMENTO ; Ainda nao...

205 RETURN ; Sim

206

207 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

208 ; CALCULA

209 ; Retorna em DIF a maneira otima de chegar a posicao final

210 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

211 CALC:

212 MOVF POSITION, W

213 SUBWF TARGET, W

214 MOVWF VAR1 ; VAR1 = TARGET - POSITION

215

216 BTFSS VAR1, 7 ; VAR1 e' negativo?

217 GOTO $+3

218 COMF VAR1, W

219 ADDLW .1 ; W = - VAR1

220

221 SUBLW PASSOS\_2 ; W = PASSOS/2 - |VAR1|

222

223 BTFSC STATUS, C ; Deu borrow?

224 GOTO NORMAL ; Não

225 ALTERADO: ; Transicao F <-- 0 ou F --> 0

226 MOVF VAR1, W

227 BTFSS VAR1, 7

228 GOTO $+3

229 ADDLW PASSOS

230 GOTO $+2

231 ADDLW -PASSOS

232 MOVWF DIF

233 RETURN

234 NORMAL:

235 MOVF VAR1, W

236 MOVWF DIF

237 RETURN

238

239 ; DIF guarda a quantidade de passos que devera ser dada.

240 ; Caso seja negativo, estara em complemento 2

241

242 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

243 ; MOVER

244 ; Move um passo na direcao determinada por DIF

245 ; Retorna em W o estado de energizacao das bobinas

246 ; Atualiza POSITION e DIF

247 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

248 MOVER:

249 MOVF DIF, F

250 BTFSC STATUS, Z ; DIF é zero?

251 GOTO TABELA\_BOBINA ; Sim

252

253 BTFSC DIF, 7 ; DIF é negativo?

254 GOTO CELERA\_ESQUERDA ; Sim

255

256 CELERA\_DIREITA:

257 DECF DIF

258

259 INCF POSITION

260 MOVLW PASSOS

261 XORWF POSITION, W

262 BTFSC STATUS, Z ; POSITION = PASSOS?

263 CLRF POSITION ; Sim, Entao Zera POSITION!

264

265 INCF BOBINA\_8

266 MOVLW .8

267 XORWF BOBINA\_8, W

268 BTFSC STATUS, Z ; BOBINA\_8 = 8?

269 CLRF BOBINA\_8 ; Sim, Entao Zera BOBINA\_8

270 GOTO TABELA\_BOBINA

271

272 CELERA\_ESQUERDA:

273 INCF DIF

274

275 MOVF POSITION

276 BTFSS STATUS, Z ; POSITION = 0 ?

277 GOTO $+3 ; Nao. So decrementa

278 MOVLW PASSOS ; Sim. Reinicia POSITION e decrementa

279 MOVWF POSITION

280 DECF POSITION ; de 0 a 2\*n-1

281

282 MOVF BOBINA\_8

283 BTFSS STATUS, Z ; BOBINA\_8 = 0 ?

284 GOTO $+3 ; Nao. So decrementa

285 MOVLW .8 ; Sim. Reinicia BOBINA\_8 e decrementa

286 MOVWF BOBINA\_8

287 DECF BOBINA\_8 ; de 0 a 7

288

289 TABELA\_BOBINA:

290 MOVF BOBINA\_8, W

291 ADDWF PCL

292 RETLW B'00010000' ; BOBINA\_8 = 0

293 RETLW B'00110000' ; BOBINA\_8 = 1

294 RETLW B'00100000' ; BOBINA\_8 = 2

295 RETLW B'01100000' ; BOBINA\_8 = 3

296 RETLW B'01000000' ; BOBINA\_8 = 4

297 RETLW B'11000000' ; BOBINA\_8 = 5

298 RETLW B'10000000' ; BOBINA\_8 = 6

299 RETLW B'10010000' ; BOBINA\_8 = 7

300

301 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

302 ; Rotinas de DELAY

303 ; DELAY\_FLAG é o tempo de espera adicional quando o motor chega a uma posição determinada

304 ; DELAT\_PASSO é o tempo adicional de espera entre os passos para energizar as bobinas

305 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

306 DELAY\_FLAG:

307 ;2999995 cycles

308 movlw 0x1A

309 movwf VAR1

310 movlw 0x8B

311 movwf VAR2

312 movlw 0x07

313 movwf VAR3

314 DELAY\_FLAG\_0

315 decfsz VAR1, f

316 goto $+2

317 decfsz VAR2, f

318 goto $+2

319 decfsz VAR3, f

320 goto DELAY\_FLAG\_0

321

322 ;1 cycle

323 nop

324

325 ;4 cycles (including call)

326 return

327 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

328 DELAY\_PASSO:

329 ;4993 cycles

330 movlw 0xE6

331 movwf VAR1

332 movlw 0x04

333 movwf VAR2

334 DELAY\_PASSO\_0

335 decfsz VAR1, f

336 goto $+2

337 decfsz VAR2, f

338 goto DELAY\_PASSO\_0

339

340 ;3 cycles

341 goto $+1

342 nop

343

344 ;4 cycles (including call)

345 return

346 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

347

348 END

281

282 MOVF BOBINA\_8

283 BTFSS STATUS, Z ; BOBINA\_8 = 0 ?

284 GOTO $+3 ; Nao. So decrementa

285 MOVLW .8 ; Sim. Reinicia BOBINA\_8 e decrementa

286 MOVWF BOBINA\_8

287 DECF BOBINA\_8 ; de 0 a 7

288

289 TABELA\_BOBINA:

290 MOVF BOBINA\_8, W

291 ADDWF PCL

292 RETLW B'00010000' ; BOBINA\_8 = 0

293 RETLW B'00110000' ; BOBINA\_8 = 1

294 RETLW B'00100000' ; BOBINA\_8 = 2

295 RETLW B'01100000' ; BOBINA\_8 = 3

296 RETLW B'01000000' ; BOBINA\_8 = 4

297 RETLW B'11000000' ; BOBINA\_8 = 5

298 RETLW B'10000000' ; BOBINA\_8 = 6

299 RETLW B'10010000' ; BOBINA\_8 = 7

300

301 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

302 ; Rotinas de DELAY

303 ; DELAY\_FLAG é o tempo de espera adicional quando o motor chega a uma posição determinada

304 ; DELAT\_PASSO é o tempo adicional de espera entre os passos para energizar as bobinas

305 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

306 DELAY\_FLAG:

307 ;2999995 cycles

308 movlw 0x1A

309 movwf VAR1

310 movlw 0x8B

311 movwf VAR2

312 movlw 0x07

313 movwf VAR3

314 DELAY\_FLAG\_0

315 decfsz VAR1, f

316 goto $+2

317 decfsz VAR2, f

318 goto $+2

319 decfsz VAR3, f

320 goto DELAY\_FLAG\_0

321

322 ;1 cycle

323 nop

324

325 ;4 cycles (including call)

326 return

327 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

328 DELAY\_PASSO:

329 ;4993 cycles

330 movlw 0xE6

331 movwf VAR1

332 movlw 0x04

333 movwf VAR2

334 DELAY\_PASSO\_0

335 decfsz VAR1, f

336 goto $+2

337 decfsz VAR2, f

338 goto DELAY\_PASSO\_0

339

340 ;3 cycles

341 goto $+1

342 nop

343

344 ;4 cycles (including call)

345 return

346 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

347

348 END

### PCB – Placa de Circuito Impresso – PIC de Controle

Foi confeccionada uma placa para o PIC de controle, nela temos um PIC16F628A, 9 resistores (8 de 330 Ω e 1 de 10kΩ), 2 barras de pinos e display de 7 segmentos de Anodo Comum. Abaixo temos a Figura 20 com a PCB do PIC de controle, e abaixo dela a Figura 21 com Simulação 3D do Proteus e Figura 22 com placa física.

Figura - PCB PIC de Controle

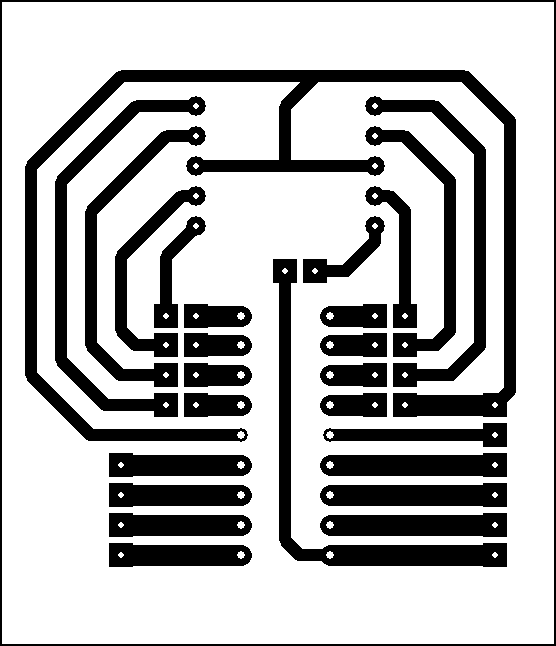
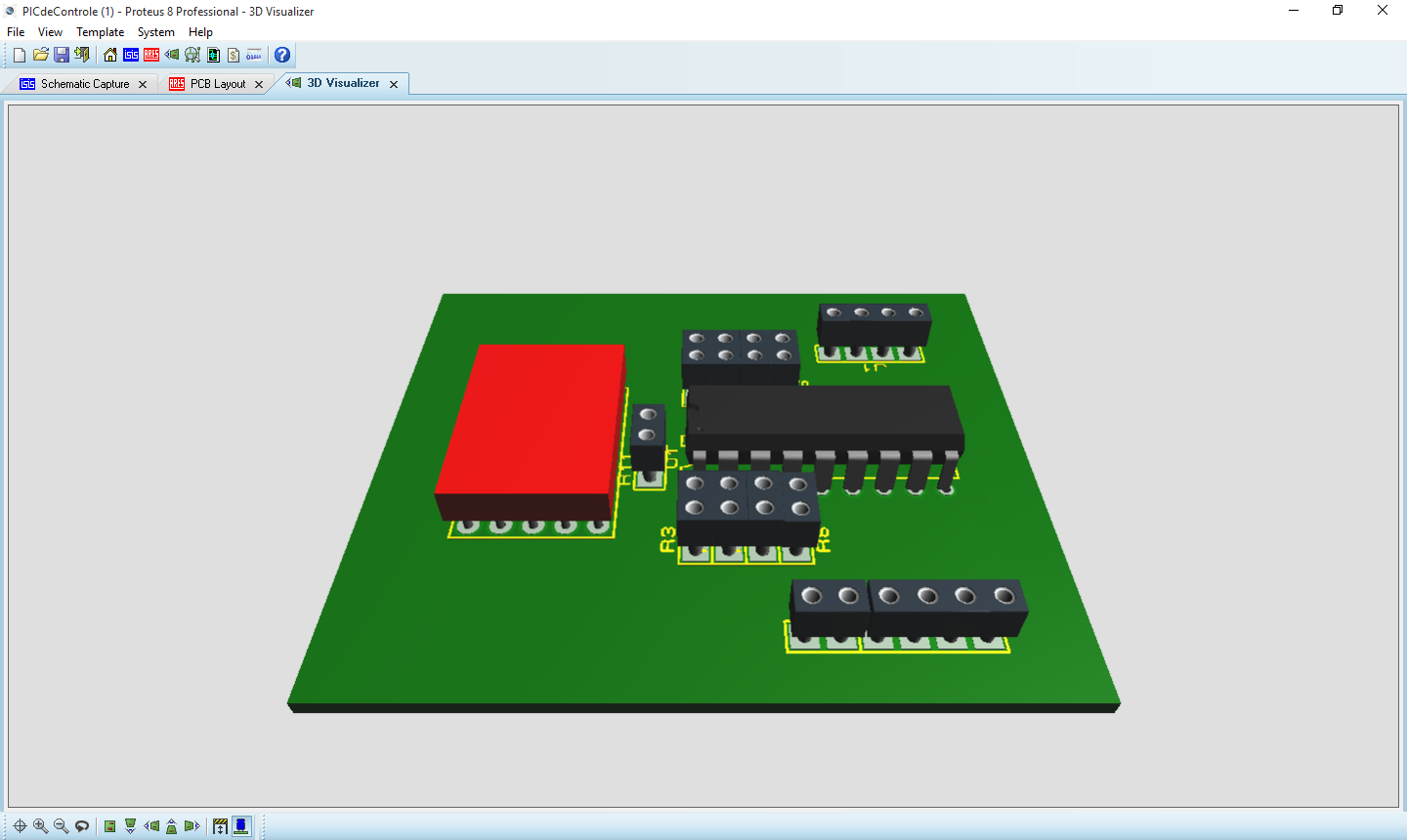
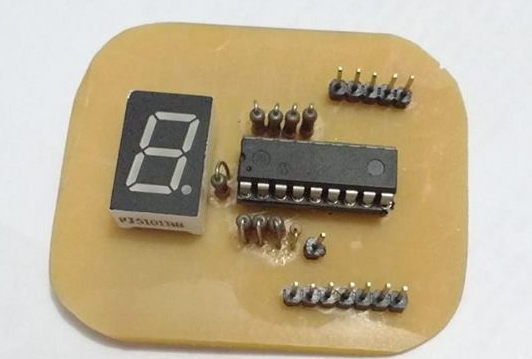


Figura - Placa Simulada em 3D no Proteus



Placa física:

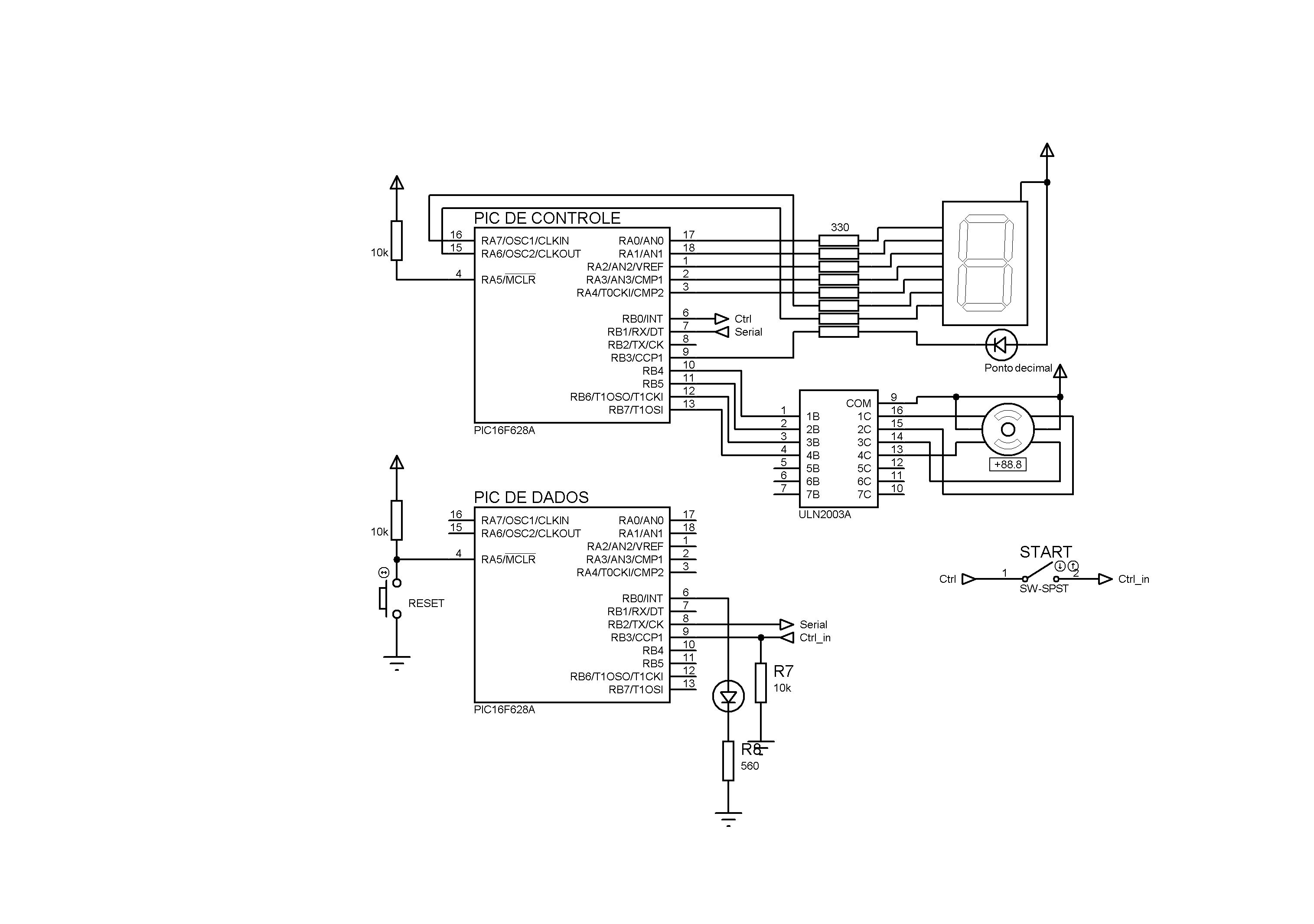
Figura - Placa Física PIC de Controle



### Esquemático do Projeto – Isis Proteus

A Figura 23 representa o esquemático final do projeto realizado. Esse esquemático foi desenhado e inclusive simulado no mesmo programa, tendo em vista que o programa possui um ótimo recurso de simulação.

Figura - Esquemático do Projeto no Proteus



# Conclusão

Diante de todas as informações constituintes deste relatório e das análises práticas obtidas durante a construção do projeto, conclui-se que esse foi executado com sucesso. Já que, em seus últimos testes, o projeto funcionou de acordo com o planejado, assim, espera-se o mesmo comportamento durante a sua exposição na avaliação prática.

No entanto, para a realização deste feito, deve-se ressaltar o aprimoramento da lógica de programação utilizada, uma vez que a otimização do projeto foi devidamente considerada pela equipe realizadora. Com isso, superaram-se problemas com a movimentação da haste, a qual deve girar a favor do sentido (horário ou anti-horário) mais próximo do caractere seguinte a ser apontado - menor número de passos. Além disso, dificuldades com a determinação das informações do motor – como sua resistência elétrica, sua corrente elétrica necessária para a devida alimentação e seu número de passos – foram contornadas mediante a realização de testes. Ademais, para evitar possíveis problemas por mau contato dos componentes, a confecção da placa foi considerada essencial.

Dado o exposto, o presente relatório descreve uma das formas efetivas para a correta construção do projeto exigido pelos professores de Engenharia Elétrica da Universidade Federal do Ceará na disciplina de Microprocessadores.

# Referências Bibliográficas

BRITES, F. G.; SANTOS, V. P. A. **Motor de passo**. 2008. 15 p. Disponível em: <https://www.telecom.uff.br/pet/petws/downloads/tutoriais/stepmotor/stepmotor2k81119.pdf>. Acesso em: 20 jan. 2016.

CCM Benchmark Group. **O código ASCII**. Disponível em: <http://br.ccm.net/contents/54-o-codigo-ascii>. Acesso em: 20 jan. 2016.

Ebah. **Microcontroladores**. Disponível em: <http://www.ebah.com.br/content/ABAAABr-kAG/microcontroladores>. Acesso em: 18 jan. 2016.

Microchip Technology Inc. **Datasheet***:* PIC16F627A/628A/648A. 2007.

ShenZhen Wayjun Technology Co.,Ltd. **Segment digit LED display**. 2011. 51 p. Disponível em: <https://e-radionica.com/productdata/LD3361BS.pdf>. Acesso em: 23 jan. 2016.

SILVA, L. M. C. **Análise de circuitos digitais - PIC16F628a**. 8 p. Disponível em: <https://vinivpns.files.wordpress.com/2010/04/pic16f628a-comandos-de-programa.pdf>. Acesso em: 18 jan. 2016.