TUTORIAL
ENTRADAS DIGITAIS

Autores: Luís Fernando Patsko e Tiago Lone
Nível: Intermediário
Criação: 27/12/2005
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

Nesse tutorial vamos aprender a utilizar as portas de entradas digitais presentes no *Módulo de Entradas, Saídas e Servo-Motores* do KDR5000. Esse módulo possui duas dessas portas, sendo que cada uma delas possui 8 bits. Através desse programa também é possível utilizar a porta de entradas digitais do MEC1000. Para auxiliar nas explicações vamos criar um programa que permite ler o estado dessas portas de entrada e veremos também um pouco sobre como trabalhar com elas do ponto de vista eletrônico. Dessa forma teremos total controle sobre elas e poderemos usá-las sem problemas em nossos projetos.

2 – Material

Para esse tutorial é necessário o *Módulo Principal* e o *Módulo de Entradas, Saídas e Servo-Motores*. Para a criação do programa será necessário o Borland Delphi 6. A seguir a imagem da montagem do Kit necessária para esse tutorial.

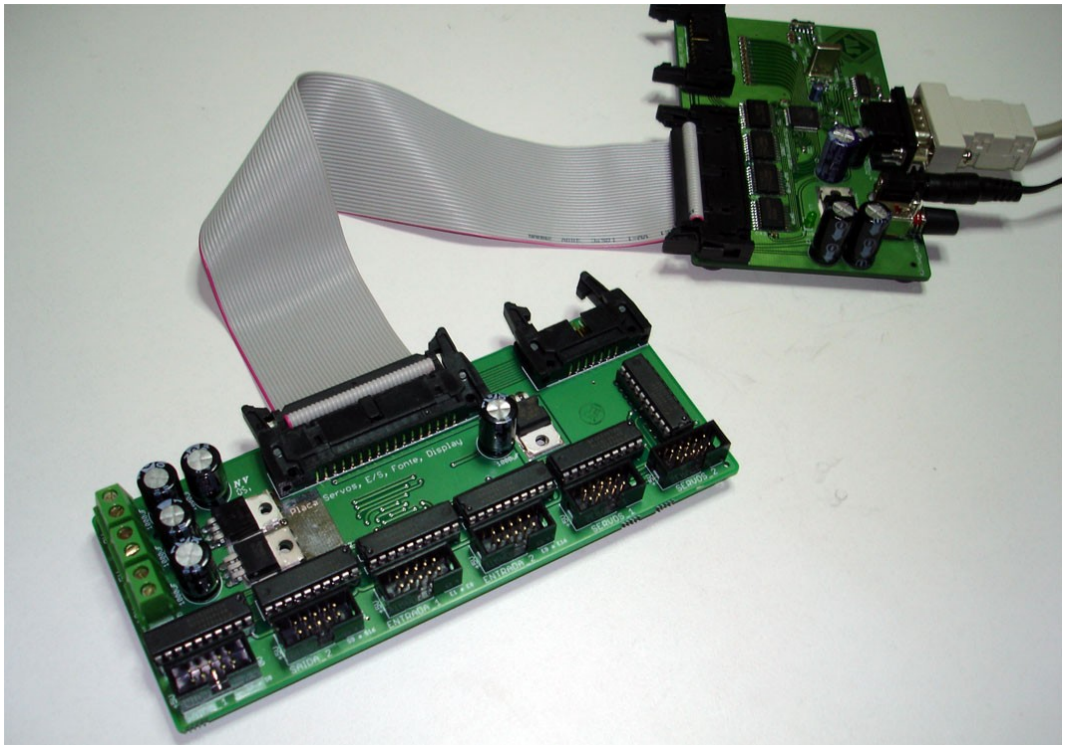

Figura 1: *Montagem do kit usada nesse tutorial.*

3 – Projeto

Nesse tutorial vamos desenvolver um programa para ler o estado das portas de entrada digitais do *Módulo de Entradas, Saídas e Servo-Motores*. Com ele poderemos visualizar o estado de todos os bits das portas. A aparência desse programa será a seguinte.

Figura 2: *Aparência final do programa que será criado nesse tutorial.*

Nosso primeiro passo é criar essa interface gráfica. 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.

Essa interface possui um Label, um ComboBox, oito Shape e um Button. O ComboBox serve para selecionar a porta que iremos ler, os Shape indicam o estados dos bits da porta e o Button atualiza os Shape fazendo uma nova leitura da porta. O componente Label, o ComboBox e o Button encontram-se na aba "Standard" da barra de componentes, já o Shape encontra-se na aba "Additional".

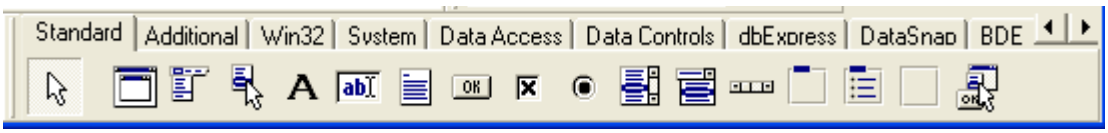

Figura 3: *Aba "Standard" da Barra de componente.*

Vamos adicionar um Label e um ComboBox para a seleção da porta que iremos ler o estado. 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.*

Adicionamos o Label e modificamos as seguintes propriedades.

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

Em seguida adicionamos um componente ComboBox e modificamos as propriedades a seguir.

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

Com isso nosso Form irá se parecer com o seguinte.

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

Vamos adicionar os componentes Shape que irão nos informar o estado da porta. Esse componente encontra-se na aba “Additional” da barra de componentes.

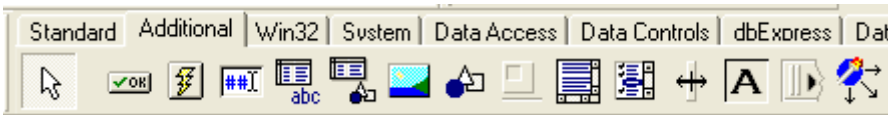

Figura 7: *Aba "Additional" da barra de componentes.*

Ele possui o seguinte ícone.

Figura 8: *Ícone do componente Shape.*

Vamos adicionar oito desses componentes e modificar as seguintes propriedades de todos eles. A seguir são apresentadas as propriedades apenas dos dois primeiros, os outros seguem o mesmo estilo, sendo que, apenas o nome irá mudar.

Name	=	ShapeBit0
Width	=	20
Height	=	20
Brush/Color	=	clGreen
Shape	=	stCircle

Name	=	ShapeBit1
Width	=	20
Height	=	20
Brush.Color	=	clGreen
Shape	=	stCircle

Assim teremos o Form da seguinte forma.

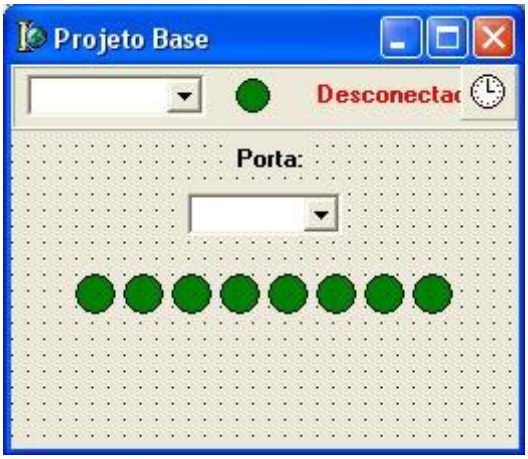

Figura 9: *Aparência do Form após a inclusão dos componentes Shape.*

Nesse ponto só falta adicionar o botão para atualização do estado dos Shape de acordo com a leitura da porta. Para isso adicionamos um componentes Button que pode ser encontrado na aba "Standard" da barra de componentes. Esse componente possui o seguinte ícone.

Figura 10: *Ícone do componente Button.*

Temos que modificar as seguintes propriedades do botão.

Name	=	ButtonAtualizar
Caption	=	Atualizar
Font/Style/fsBold	=	true

Um último detalhe que vamos modificar é a propriedade Caption do Form principal. Como copiamos o projeto do tutorial Base, essa propriedade possui o valor "Projeto Base". Vamos modificar essa propriedade para "Entradas Digitais". Com isso finalizamos a construção de nossa interface gráfica. A seguir a imagem dessa interface finalizada.

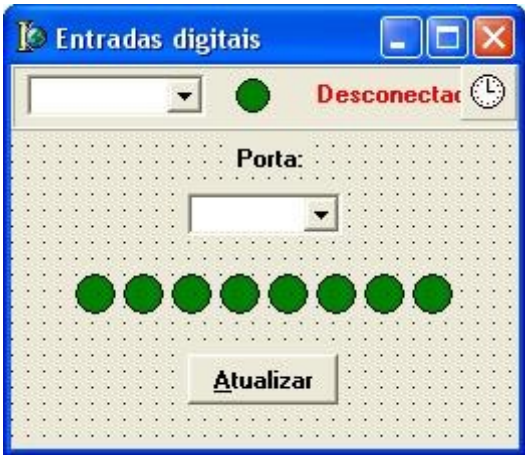

Figura 11: *Interface finalizada.*

Temos agora que criar o código de leitura da porta e atualização da interface gráfica. Tanto a leitura da porta quanto a atualização da interface serão executadas quando o botão "Atualizar" for pressionado. Logo o código para essas operações estará dentro do manipulador do evento OnClick do botão "Atualizar".

Para criar esse manipulador podemos selecionar o componente Button que possui o texto "Atualizar", ir no **Object Inspector**, selecionar a aba Events e dar um duplo clique sobre a linha que está escrito OnClick. Uma forma mais fácil de fazer isso é apenas dar um duplo clique sobre o botão no Form e com isso o Delphi irá criar automaticamente um manipulador para o evento OnClick. O seguinte código será criado.

```
Procedure TformMain.ButtonAtualizarClick(
                                        Sender: TObject);
begin
end;
```

Vamos primeiro adicionar dentro desse manipulador o código que irá fazer a leitura do estado da porta. O método que vamos utilizar para fazer essa leitura é o DigitalPortRead. Esse método possui a seguinte declaração.

```
Procedure DigitalPortRead(port : Integer) : Byte;
```

Ele possui apenas um parâmetro que indica que porta que queremos ler. O *Módulo de Entradas, Saídas e Servo-Motores* possui duas portas de entradas digitais, logo o parâmetro desse método pode ser "0", para indicar a primeira porta, ou "1", para indicar a segunda.

Esse método retorna um byte com o estado da porta lida. Cada bit desse retorno irá indicar o estado de um bit da porta de entradas digitais. O bit 0 da porta será representado pelo bit 0 do valor retornado, o bit 1 pelo bit 1 e assim por diante. Para maiores informações sobre o funcionamento e uso das portas de entradas digitais veja o tópico sobre essas portas ainda nesse

documento.

Agora vamos iniciar a implementação do nosso manipulador do evento OnClick do botão “Atualizar”. O início do código ficará assim.

```
Procedure TformMain.ButtonAtualizarClick(
                                    Sender: TObject);
var
  // Armazena valor lido da porta
  bits : Byte;

begin
  // Lê porta
  bits := kit.DigitalPortRead(
                          ComboBoxPorta.ItemIndex);

  ...
end;
```

Nesse início do código apenas declaramos a variável denominada “bits” e em seguida armazenamos nela o retorno do método DigitalPortRead, ou seja, o estado da porta que está selecionada no ComboBox de seleção. Como parâmetro desse método utilizamos a propriedade ItemIndex do ComboBox, que será “0” se a porta selecionada for a primeira e “1” caso a porta selcionada seja a segunda.

Com o estado da porta armazenado na variável denominada “bits”, temos que atualizar a interface gráfica para refletir esse estado. Vamos verificar bit a bit o estado da porta e modificaremos a cor do Shape correspondente aos bits de acordo com seu estado. Fazemos isso com o seguinte código.

```
  ...

  // Lê porta
  bits := kit.DigitalPortRead(
                          ComboBoxPorta.ItemIndex);

  // Atualiza interface
  // Bit 0
```

```
if (bits AND 1) <> 0 then
  ShapeBit0.Brush.Color := clLime
else
  ShapeBit0.Brush.Color := clGreen;

// Bit 1
if (bits AND 2) <> 0 then
  ShapeBit1.Brush.Color := clLime
else
  ShapeBit1.Brush.Color := clGreen;

// Bit 2
if (bits AND 4) <> 0 then
  ShapeBit2.Brush.Color := clLime
else
  ShapeBit2.Brush.Color := clGreen;

// Bit 3
if (bits AND 8) <> 0 then
  ShapeBit3.Brush.Color := clLime
else
  ShapeBit3.Brush.Color := clGreen;

// Bit 4
if (bits AND 16) <> 0 then
  ShapeBit4.Brush.Color := clLime
else
  ShapeBit4.Brush.Color := clGreen;

// Bit 5
if (bits AND 32) <> 0 then
  ShapeBit5.Brush.Color := clLime
else
  ShapeBit5.Brush.Color := clGreen;
```

```
    // Bit 6
    if (bits AND 64) <> 0 then
      ShapeBit6.Brush.Color := clLime
    else
      ShapeBit6.Brush.Color := clGreen;

    // Bit 7
    if (bits AND 128) <> 0 then
      ShapeBit7.Brush.Color := clLime
    else
      ShapeBit7.Brush.Color := clGreen;
end;
```

Em cada bloco "if" verificamos se o bit que o bloco testa está setado para "1". Fazemos isso utilizando a operação lógica AND. Se o bit que estivermos testando estiver setado para "1", então o resultado da operação AND, da variável "bits" com o valor para testar o bit, será diferente de zero. A tabela a seguir mostra o valor que temos que utilizar, em conjunto com a operação AND, para testar um determinado bit.

Bit Testado	*Valor para Teste*	*Resultado quando Bit é "1"*	*Resultado quando Bit é "0"*
0	1	1	0
1	2	2	0
2	4	4	0
3	8	8	0
4	16	16	0
5	32	32	0
6	64	64	0
7	128	128	0

Por essa tabela, se queremos testar se o bit de número 2 de um valor é igual a "1" ou "0", então fazemos a operação lógica AND desse valor com o número 4 e se o resultado for diferente de 0 então o bit será "1" senão será "0". Quatro em binário é 00000100b, ou seja, um valor em que apenas seu bit de número 2 é "1". Se quiséssemos testar o bit de número 1 então utilizaríamos o valor 2, em binário 000000010b. Para testar o bit de número 0 utilizaríamos o valor 1, que em binário é 00000001b. Para maiores informações sobre a operação lógica AND procure alguma documentação sobre operações lógicas.

Nesse momento já podemos testar o programa. Assim que o executamos, selecionamos a porta serial correta e pressionamos o botão "Atualizar", a interface gráfica será atualizada para refletir o estado da porta de entrada selecionada.

A seguir algumas imagens do programa após a leitura do estado da primeira porta de entradas digitais.

Figura 13: *Leitura da porta com o estado setado para 10100101b.*

Figura 12: *Leitura da porta com todos os bits setados para "1", ou seja, 11111111b.*

Figura 14: *Leitura da porta com o estado setado para 11110000b.*

Com isso temos um programa que lê o estado das portas e o apresenta de forma gráfica. A seguir vamos ver um pouco mais sobre as portas de entradas digitais e como podemos utilizá-las em projetos práticos.

4 – Portas de Entradas digitais

As portas de entradas digitais foram projetadas para uso genérico, possibilitando ao usuário a conexão de qualquer circuito ao KDR5000 ou ao MEC1000 para fazer uso de suas

funcionalidades. Sua versatilidade e flexibilidade permitem inúmeras combinações e, com o devido conhecimento de eletrônica e programação, as possibilidades de utilização são ilimitadas.

O usuário apenas deverá respeitar os limites quanto a configuração das entradas digitais. Atenção extra deverá ser aplicada no que diz respeito à capacidade e à pinagem destas. Os circuitos a serem conectados deverão suportar convenientemente a tensão de 5V proveniente do pino 1 do conector e os sinais a serem aplicados às entradas não podem superar 5V. São detalhes importantes que deverão ser respeitados, caso contrário, danos podem ser ocasionados a ambos os equipamentos. É importante notar que a entrada não deve ser ligada diretamente à tensão de 5V, mas através de um resistor, de modo a limitar a corrente.

A pinagem é também algo ao qual deve ser tomado muito cuidado. Qualquer descuido também poderá resultar em danos. A imagem a seguir mostra a disposição dos conectores das entradas digitais.

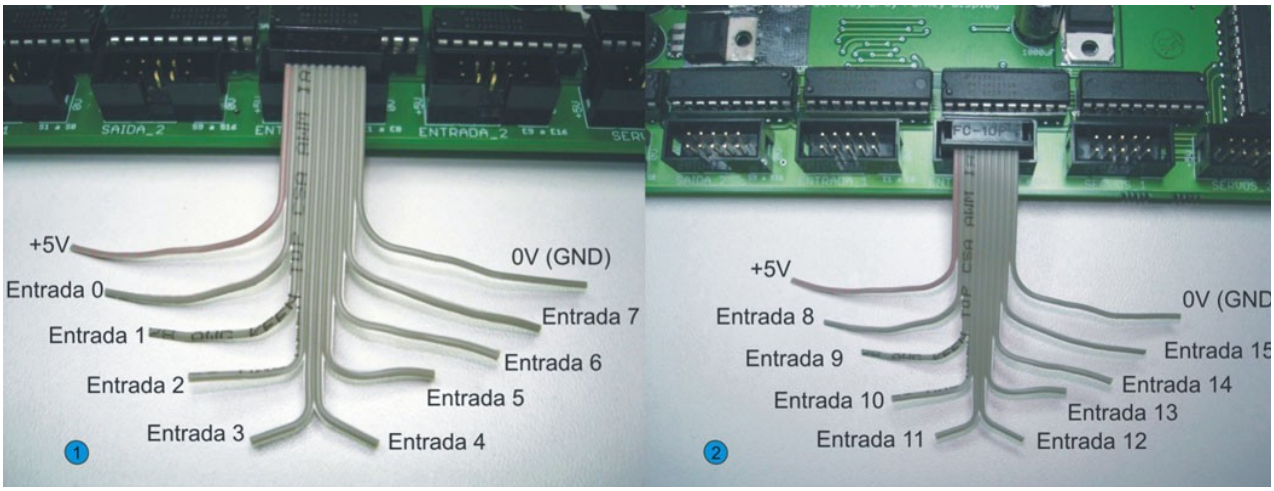

Figura 15: *Configuração das entradas digitais: 1-Porta de Entradas Digitais 0, 2-Porta de Entrada Digitais 1.*

Crimpando um cabo flat e abrindo-o, podemos compreender muito bem a pinagem dos conectores. Como pode ser observado, os conectores de alimentação estão nas extremidades em cada uma delas. Seu uso não é obrigatório caso o circuito a ser conectado já possua alimentação, entretanto deve-se tomar muito cuidado para evitar algum curto-circuito. Os fios disponíveis para a conexão com circuitos externos estão localizados no meio do cabo flat, sendo que cada um corresponde a um bit de uma entrada digital. Por exemplo, a Entrada 4 corresponde ao bit de número 4 da porta 0 e a Entrada 10 corresponde ao bit de número 2 da porta 1.

Em relação às portas de entradas digitais, a única diferença presente entre o KDR5000 e o MEC1000 é a sua quantidade. Enquanto que o KDR5000 tem a disposição do usuário 2 portas, presentes no *Módulo de Entradas, Saídas e Servo-Motores,* o MEC1000 possui apenas uma. O funcionamento e a pinagem de ambas as portas são idênticas.

Um exemplo simples que pode ser utilizado com as portas de entradas digitais é o circuito de 8 chaves mostradas a seguir. Ao acionarmos uma chave, o nível de tensão na entrada digital a qual está conectada será de 0V, fazendo com que o bit correspondente seja setado para 0. Ao abrir a chave, o nivel de tensão subirá e então o bit correspondente será setado para 1. É importante ressaltar que um resistor deve ser utilizado em cada porta, de modo a controlar a corrente que será conduzida à entrada digital.

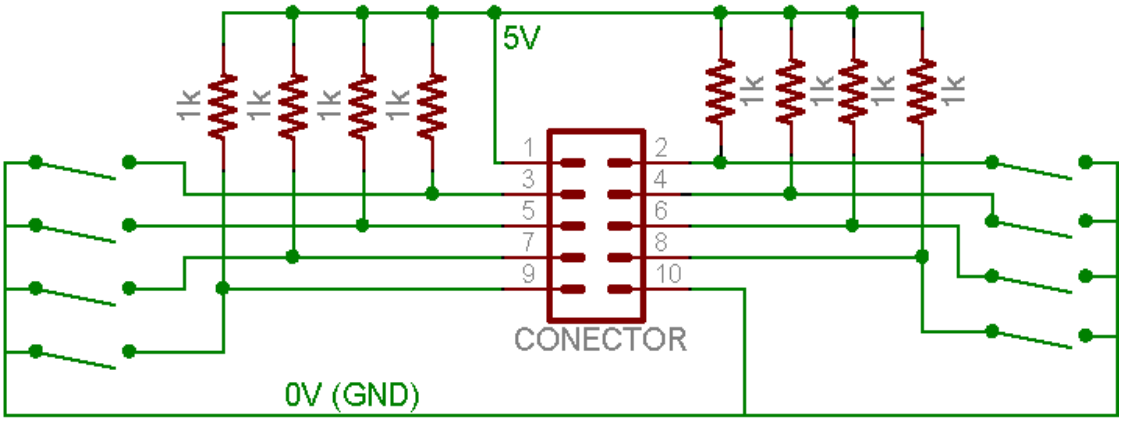

Figura 16: *Esquema de ligação de oito chaves em uma porta de entrada digital.*

Com um circuito desses e um pouco de conhecimento de programação, é possível fazer relações interessantes entre essas chaves e os outros componentes disponíveis no Kit.

5 – Dicas

Uma funcionalidade interessante que pode ser acrescentada a esse programa é a capacidade de verificar continuamente o estado das portas de entrada sem que seja necessário pressionar o botão "Atualizar".

Para essa finalidade, podemos utilizar o componente Timer, o qual já foi adicionado anteriormente no Projeto Base. Esse componente pode ser configurado para executar um método em intervalos pré-determinados. No nosso Projeto Base, ele foi configurado para verificar continuamente a presença de um Kit conectado à porta serial. Dê um duplo clique sobre ele, para exibir o código que foi criado anteriormente.

```
procedure TFormMain.TimerCheckTimer(Sender: TObject);
begin
  // Verifica se há um Kit respondendo
  if (kit.IsConnected) then
  begin
    // "Acende" LED. Muda cor de fundo do Shape
    ShapeLED.Brush.Color := clLime;

    // Muda texto para indicar conexão
    LabelConnected.Caption := 'Conectado';

    // Muda cor do texto para verde
```

```
    LabelConnected.Font.Color := clGreen;
  end
  ...
```

Devemos então configurar o Timer de modo que ele chame o método DigitalPortRead constantemente. Como o método ButtonAtualizarClick, que é manipulador do evento "On Click" do botão "Atualizar", já está configurado para chamar o método DigitalPortRead, basta então que o Timer seja complementado com uma linha de código para chamar esse método a cada intervalo. A seguir, o código com a função adicionada.

```
procedure TFormMain.TimerCheckTimer(Sender: TObject);
begin
  // Verifica se há um Kit respondendo
  if (kit.IsConnected) then
  begin
    // "Acende" LED. Muda cor de fundo do Shape
    ShapeLED.Brush.Color := clLime;

    // Muda texto para indicar conexão
    LabelConnected.Caption := 'Conectado';

    // Muda cor do texto para verde
    LabelConnected.Font.Color := clGreen;
    ButtonAtualizarClick(Sender);
  end
  ...
```

6 – Conclusão

Nesse tutorial vimos o que devemos fazer para ler o estado das portas de entradas digitais do *Módulo de Entradas, Saídas e Servo-Motores* do KDR5000 ou do MEC1000. Com o projeto que criamos foi possível entender o funcionamento do método DigitalPortRead e o significado de seu parâmetro. Vimos também como utilizar as portas de entradas digitais em conjunto com montagens eletrônicas em projetos práticos. Com isso já podemos utilizar essas portas em nossos projetos sem dificuldades.