TUTORIAL
BASE

Autor: Tiago Lone
Nível: Básico
Criação: 16/11/2005
Última versão: 18/12/2006

Maxwell Bohr
Instrumentação Eletrônica

PdP
Pesquisa e Desenvolvimento de Produtos

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

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

1 – Introdução

Nesse tutorial vamos criar um projeto que possa ser utilizado como base para outros programas de controle do KDR5000. Vamos aproveitar esse projeto para mostrar passo a passo o que é necessário para criar um programa de controle para o Kit Didático de Robótica. No final teremos um programa que permite a seleção da porta serial que será utilizada para se comunicar com o Kit, identificação automática da presença de um Kit na porta selecionada e todo o suporte para o envio de comandos de controle para o Kit, o que facilitará muito o desenvolvimento de outros programas, pois esse poderá ser utilizado como base. Esse programa também pode ser utilizado com o MEC1000.

2 – Material

Para esse tutorial é necessário apenas o *Módulo Principal* de um KDR5000 ou o MEC1000 conectado em uma porta serial do computador. Em todos os tutoriais também será necessário o Borland Delphi 6 para a criação dos programas.

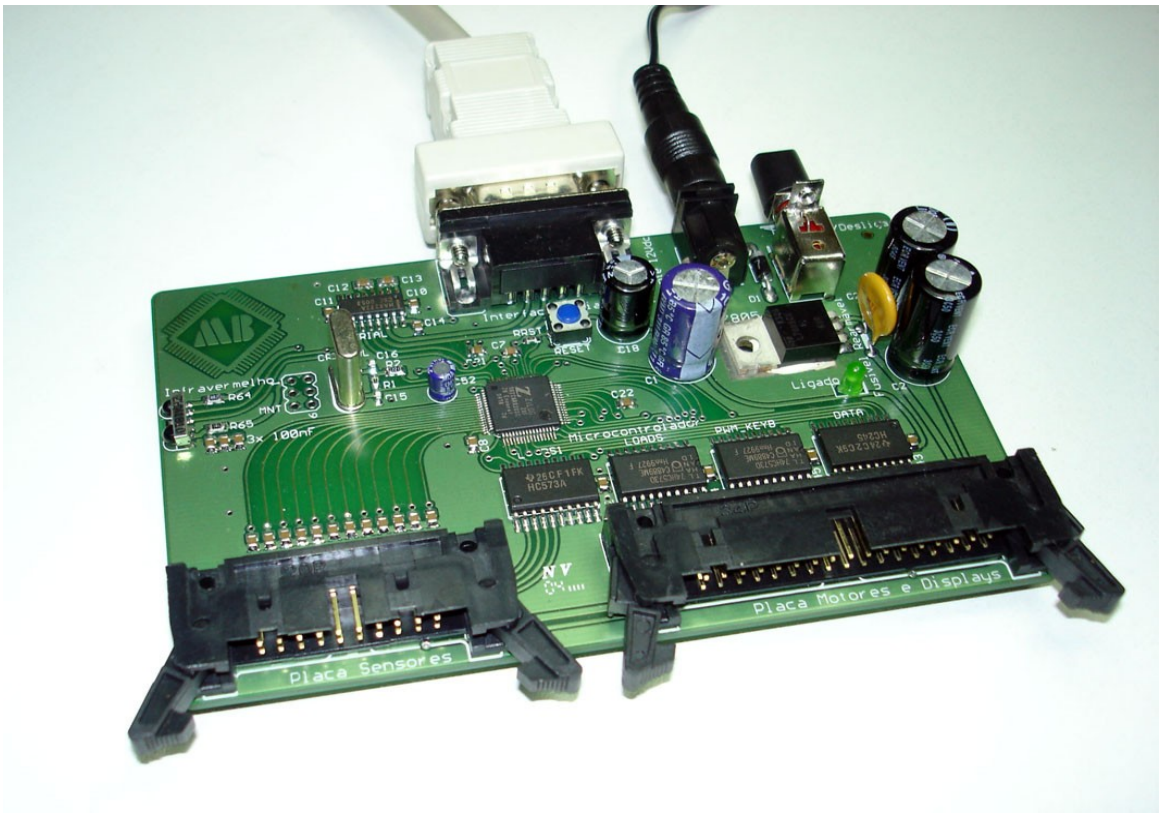

Figura 1: *Módulo Principal do KDR5000.*

3 – Projeto

O nosso primeiro passo será criar um projeto no Borland Delphi 6 que possa utilizar a biblioteca de controle do Kit Didático de Robótica. Para isso vamos abrir o Delphi 6 e criar um novo projeto. Na versão em inglês do Delphi, devemos clicar na opção de menu **File - New - Application** e com isso será criado um novo projeto de aplicativo.

Vamos salvar nosso projeto em uma pasta, por exemplo, "C:\Tutorial\Base". Para fazer isso temos que criar a pasta onde vamos salvar o projeto, em seguida ir à opção de menu **File – Save All** e indicar que queremos salvar os arquivos do projeto na pasta que criamos. Será questionado o nome de dois arquivos, o primeiro é o arquivo do Form principal, dê o nome de "Main.pas", o segundo é o arquivo de projeto, dê o nome de "Base.dpr" para ele.

Com o projeto criado devemos agora fazer com que ele possa utilizar a biblioteca de controle do Kit. Existem algumas maneiras diferentes para permitir esse uso, no entanto, usaremos aqui apenas a forma mais fácil, que é adicionar ao nosso projeto os arquivos com o código fonte da biblioteca de controle.

Para isso vamos copiar os arquivos da biblioteca de controle para a pasta do nosso projeto. Devemos copiar os arquivos com extensão ".pas" que se encontram na pasta "src" na estrutura de arquivos do projeto da biblioteca de controle. Os arquivos são "Robotica.pas", "serial\synafpc.pas", "serial\synaser.pas" e "serial\synautil.pas". Nossa pasta de projeto deverá se parecer com o seguinte:

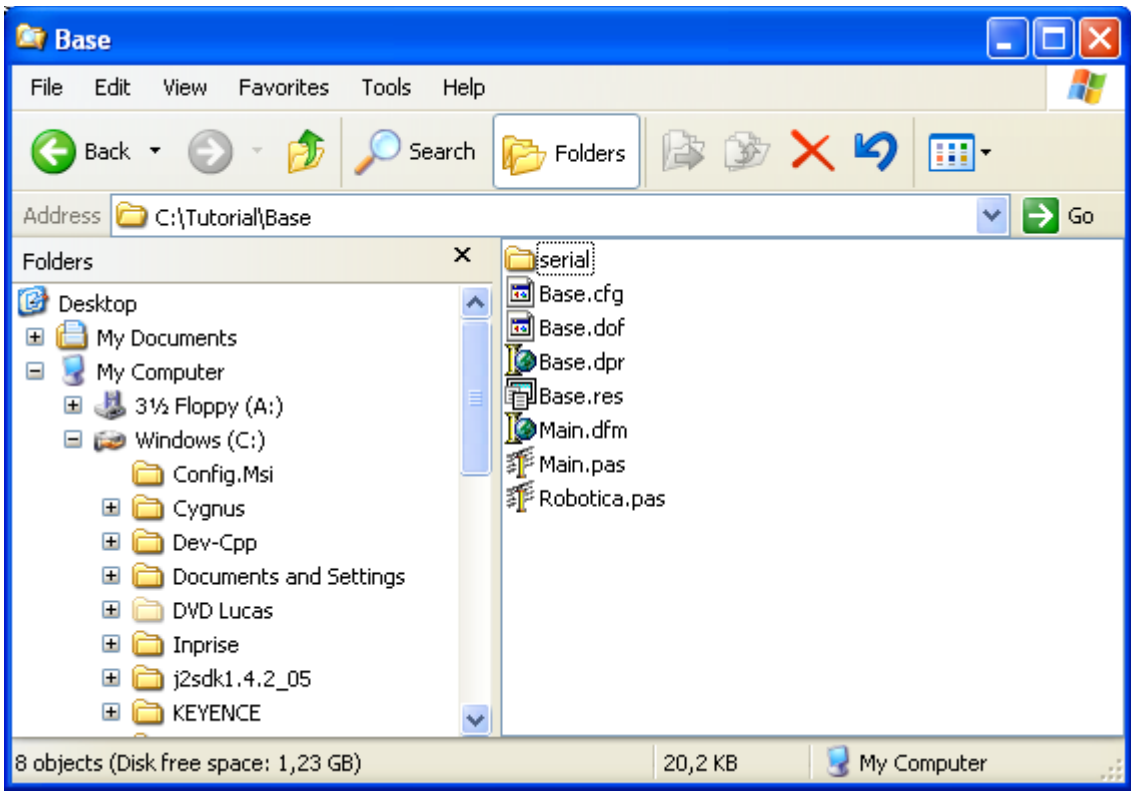

Figura 2: *Pasta do projeto Base.*

Após copiar os arquivos para a pasta do projeto, devemos incluí-los ao nosso projeto no Delphi. Para isso, vamos ao menu **Project - Add to Project...** , será apresentada uma janela em que

você deve indicar o arquivo que será incluído ao projeto. Indique um dos arquivos que você acabou de copiar para a pasta do projeto. Repita a operação até que os quatro arquivos sejam adicionados. Para verificar se está tudo correto vá à opção de menu **View - Project Manager**. Será apresentada a seguinte tela:

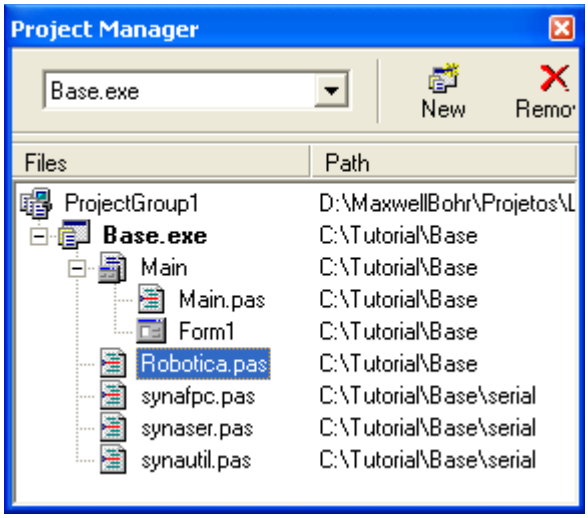

Figura 3: *Gerenciador de projeto. Podemos ver os arquivos que fazem parte do projeto.*

Nessa janela podemos ver todos os arquivos que fazem parte do projeto. Deverão ser listados os arquivos do Form principal e os quatro arquivos da biblioteca de controle, como mostra a figura. Se estiver tudo correto podemos prosseguir.

Nessa mesma janela, dê um duplo clique sobre o arquivo "Main.pas". Esse arquivo contém o código fonte do Form principal. Após o duplo clique, a seguinte tela deverá ser apresentada:

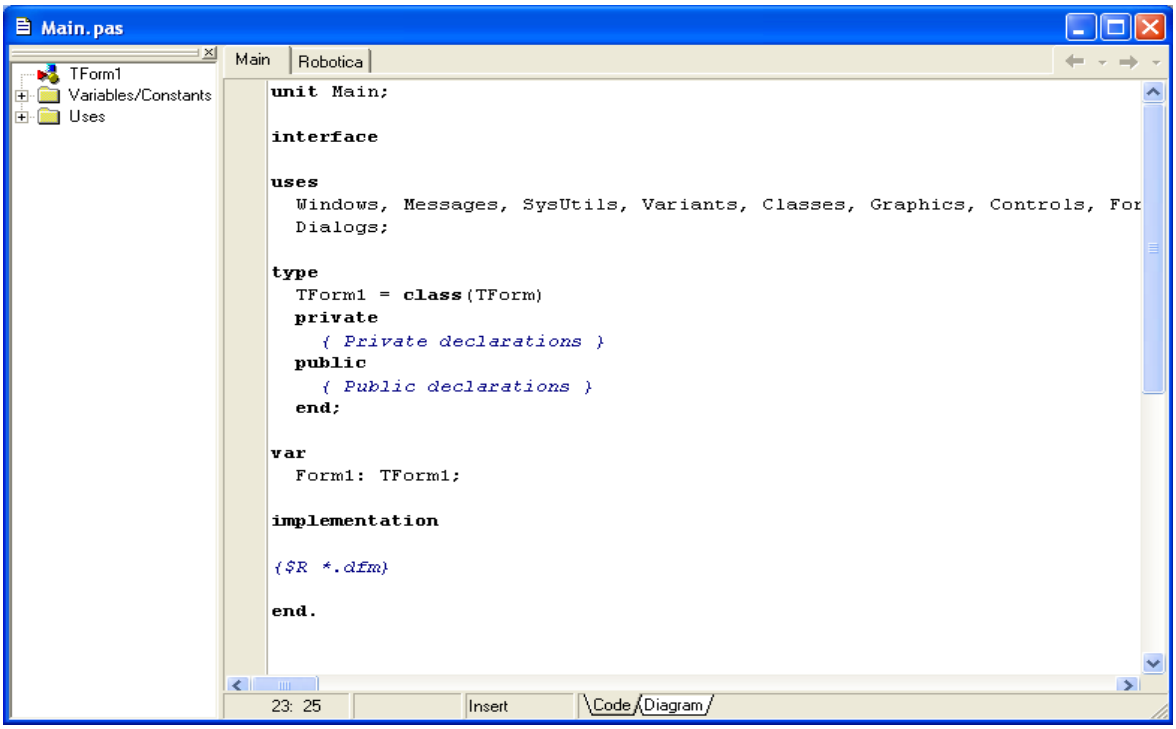

Figura 4: *Código fonte do Form principal.*

Aqui vemos o código fonte do Form principal. Para acessar as funções da biblioteca de controle temos que adicionar uma referência da Unit da biblioteca de controle do Kit em "uses". Essa Unit chama-se "Robotica". Adicione a referência à Unit "Robotica" ao final da lista de Units, como demonstrado na figura a seguir.

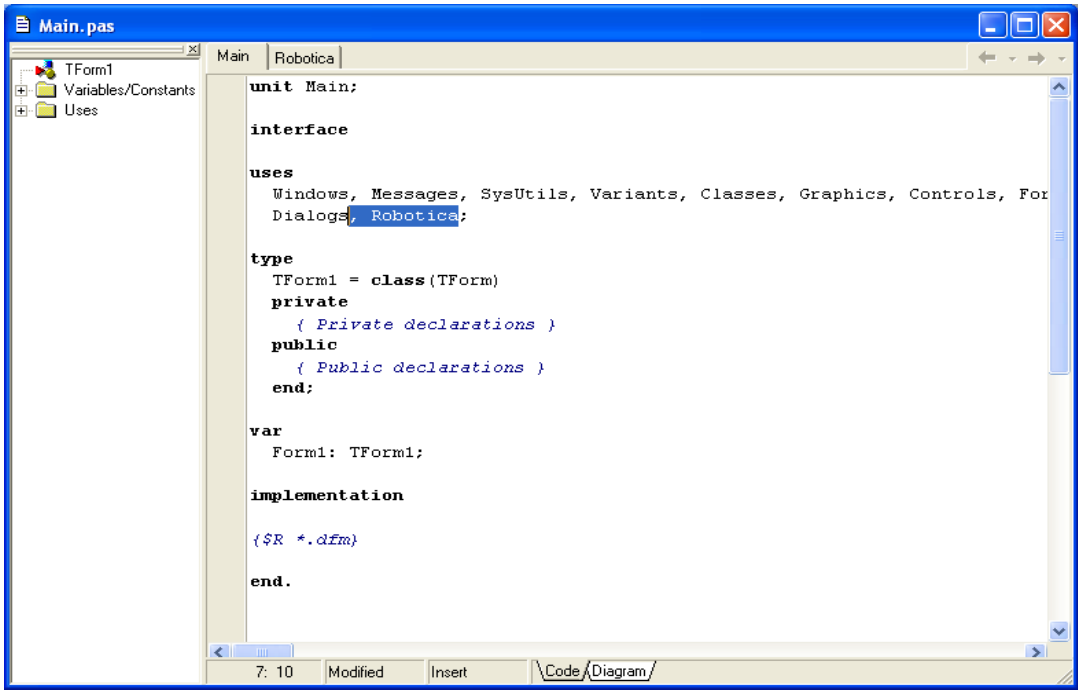

Figura 5: *Adição da referência a unit Robotica (texto selecionado).*

Agora podemos utilizar a biblioteca de controle. O principal ítem dessa biblioteca é uma classe com o nome **TKit**. Essa classe possui todos os métodos de controle do Kit. Vamos precisar de uma instância dessa classe que possa ser acessada de qualquer parte do código do Form. Para isso vamos declarar uma instância da classe **TKit** na seção **private** da classe **TForm1**. O código deverá ficar da seguinte forma:

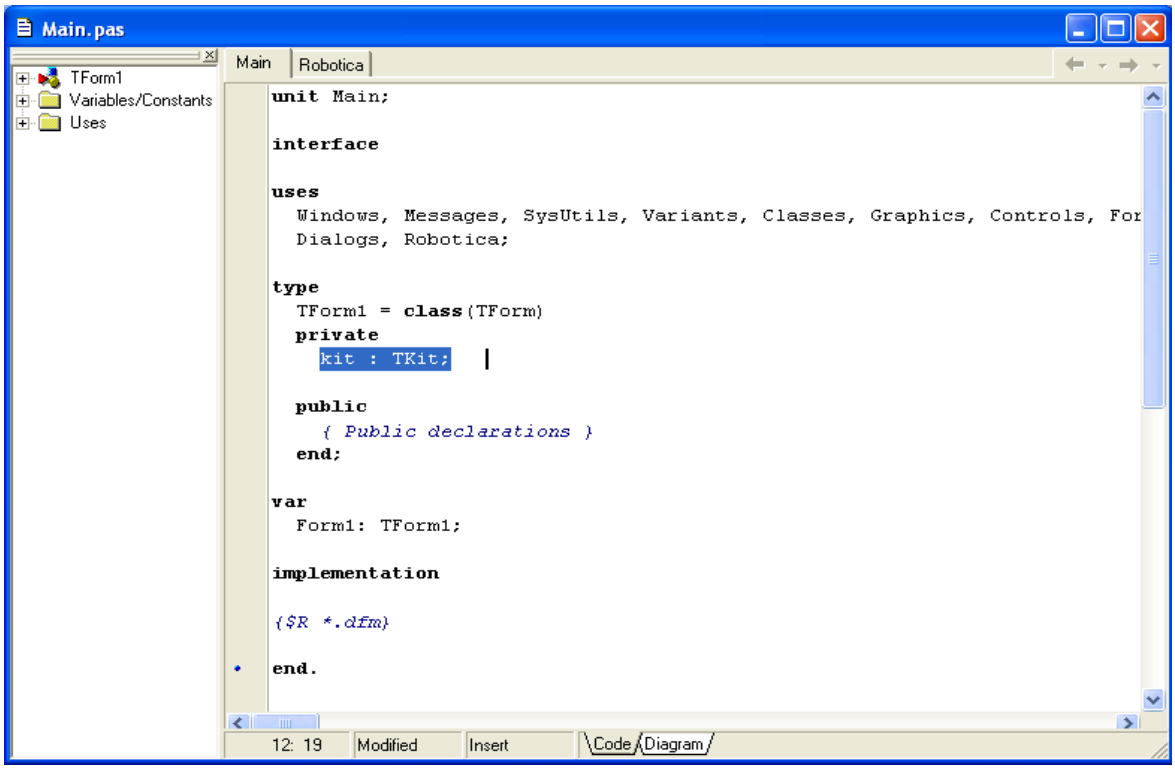

Figura 6: *Criação de uma instância da classe TKit que será utilizada pelo programa para controlar o Kit.*

Agora temos que criar a interface gráfica. Essa interface deve permitir a seleção da porta serial onde o Kit está conectado e indicar se a comunicação está correta. Para isso criaremos a interface demonstrada na figura a seguir:

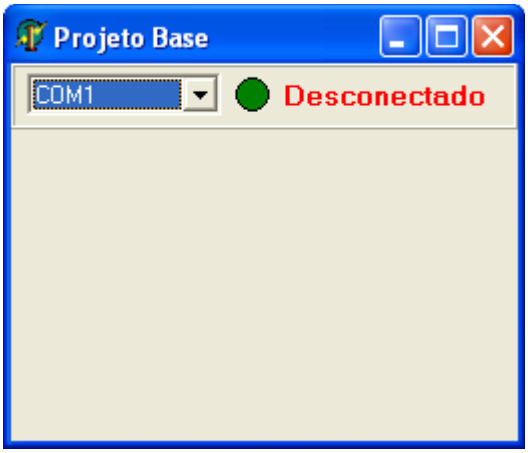

Figura 7: *Interface que permite seleção de porta serial e indica o estado da conexão com o Kit.*

Essa interface possui apenas componentes gráficos básicos. Um componente ComboBox que permite a seleção da porta serial, um componente Shape e um Label que indica o estado da conexão e um Panel para agrupar esses componentes na parte superior do programa. Vamos criar essa interface passo a passo.

Primeiramente vamos incluir um componente Panel. Esse componente pode ser encontrado na aba de componentes "Standard".

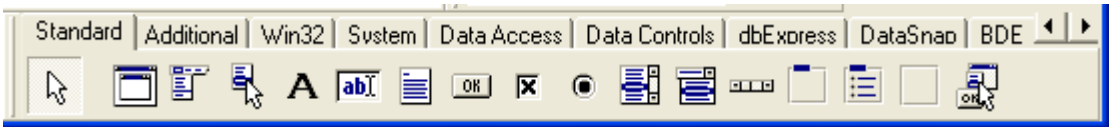

Figura 8: *Aba "Standard" da Barra de componente.*

O componente Panel possui o seguinte ícone:

Figura 9: *Ícone do componente Panel.*

Clique sobre esse ícone e em seguida clique no Form principal. Dessa forma, um componente Panel será adicionado ao Form. Agora temos que modificar algumas propriedades desse componente. Para isso, selecionamos o componente dando um clique sobre ele no Form principal. Em seguida vamos na opção de menu **View - Object Inspector** para garantir que a janela do **Object Inspector** está visível. Nessa janela podemos modificar as propriedades dos nossos componentes gráficos. Com o componente Panel selecionado podemos ver suas propriedades. A seguir uma imagem do **Object Inspector** e das propriedades de um panel.

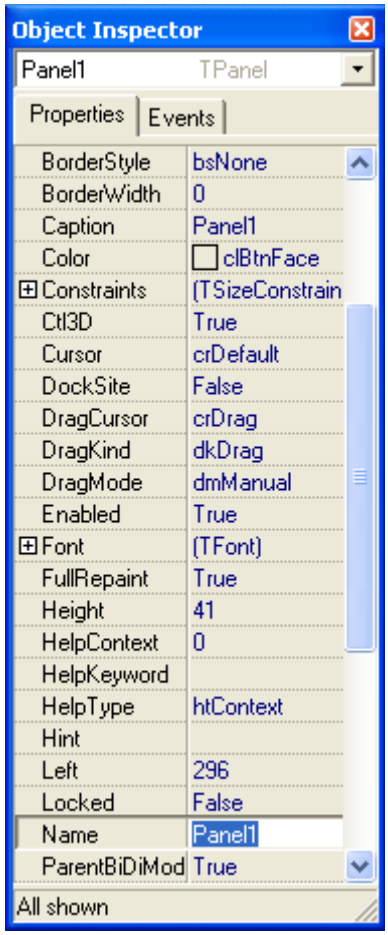

Figura 10: *Object Inspector e as propriedades do componente Panel.*

Vamos modificar as seguintes propriedades desse componente:

Name	=	PanelTop
Align	=	Top
BevelInner	=	bvLowered
Caption	=	(deixar em branco)
Height	=	33

Com isso temos uma barra superior onde vamos agrupar os outros componentes. A seguir uma imagem de como o Form principal deve estar.

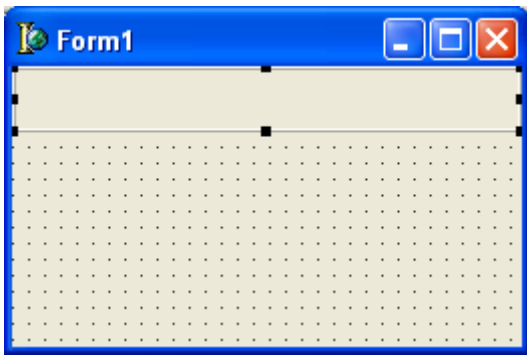

Figura 11: *Adição de um painel superior.*

O próximo passo é adicionar um ComboBox para a seleção de porta serial. Para isso selecionamos o componente ComboBox na aba "Standard" da barra de componentes e clicamos dentro do Painel superior que acabamos de adicionar. O ícone do componente ComboBox é o seguinte.

Figura 12: *Ícone do componente ComboBox.*

Após adicionar o ComboBox nosso Form terá a seguinte aparência.

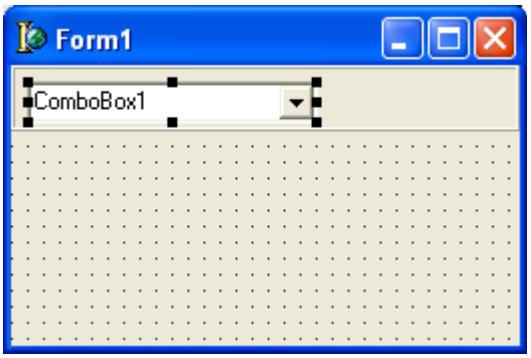

Figura 13: *Adição de um ComboBox no Painel superior.*

Vamos mudar as seguintes propriedades desse componente no **Object Inspector**:

Name	=	ComboBoxSerial
Left	=	8
Style	=	csDropDownList
Top	=	5
Width	=	100

Com o ComboBox configurado vamos adicionar dois componentes que irão indicar o estado da conexão. Um Label e um Shape. O componente Label pode ser encontrado na aba “Standard” da barra de componentes e o Shape na aba “Additional”. Esses dois componentes devem ser adicionados dentro do Painel superior. O ícone do do Label é o seguinte:

Figura 14: *Ícone do componente Label.*

A seguir a imagem da aba “Additional” da barra de componentes onde encontramos o componente Shape.

Figura 15: *Aba "Additional" da barra de componentes.*

O ícone do componente Shape é o seguinte:

Figura 16: *Ícone do componente Shape.*

A seguir uma imagem do Form principal após os componentes serem adicionados ao Painel superior.

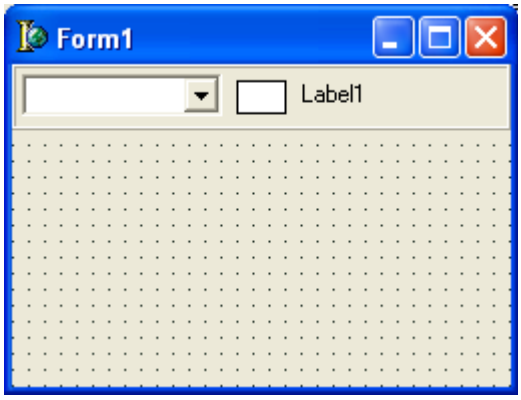

Figura 17: *Adição de um Shape e de um Label.*

Vamos mudar as seguintes propriedades do componente Shape, selecionando-o e modificando as seguintes propriedades no **Object Inspector**:

Name = ShapeLED

Brush/Color	=	clGreen
Height	=	17
Left	=	112
Shape	=	stCircle
Top	=	7
Width	=	17

No componente Label vamos mudar as seguintes propriedades:

Name	=	LabelConnected
Caption	=	Desconectado
Font/Color	=	clRed
Font/Size	=	10
Font/Style/fsBold	=	true
Left	=	136
Top	=	8

Ao final nossa barra superior irá se parecer com o seguinte:

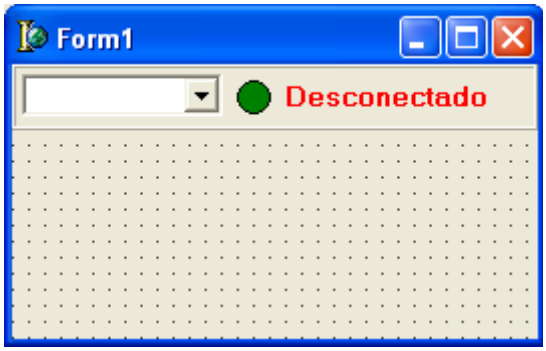

Figura 18: *Aparência final da parte visual do programa.*

Agora que temos a parte visual do programa vamos implementar suas funcionalidades. Primeiramente vamos implementar as rotinas de inicialização e finalização do programa. Podemos fazer isso incluindo código no evento OnCreate e OnDestroy do Form principal. Esses eventos são executados quando o programa inicializa e quando ele é finalizado respectivamente.

Para implementar código nesses eventos selecione o Form principal dando um clique sobre ele no **Object Tree** ou selecionando-o no **Object Inspector** diretamente. Com o Form principal selecionado o **Object Inspector** irá apresentar as propriedades do Form como mostra a figura a seguir:

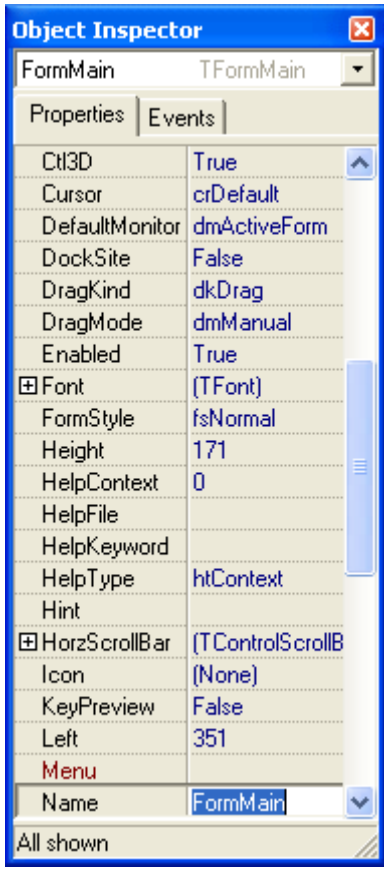

Figura 19: *Propriedades do Form principal no Object Inspector*

Vamos aproveitar e mudar as propriedades “Name” e “Caption” desse Form:

Name = FormMain

Caption = Projeto Base

Temos agora que implementar os eventos OnCreate e OnDestroy. Para isso no **Object Inspector** selecionamos, na parte superior, a aba “Events”. Serão apresentados todos os eventos do Form como podemos ver a seguir:

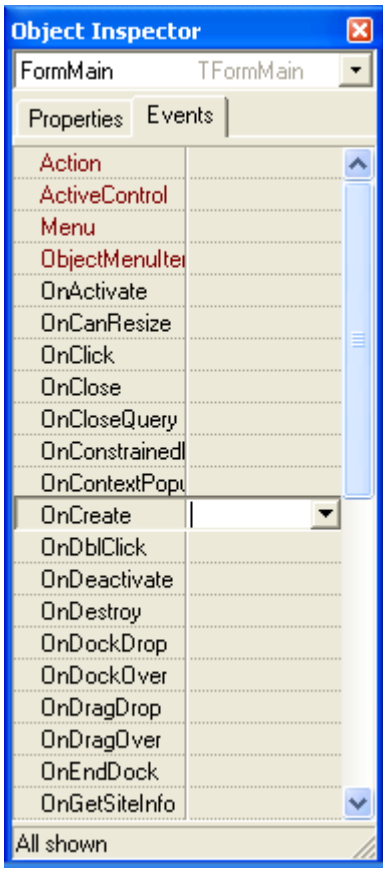

Figura 20: *Eventos do Form principal.*

Podemos visualizar nessa figura os dois eventos em que estamos interessados, OnCreate e OnDestroy. Para implementa-los é necessário apenas dar um duplo clique sobre eles e o Delphi irá automaticamente criar a função de tratamento do evento. Vamos implementar o código de inicialização no evento OnCreate. Damos um duplo clique sobre o evento e será criado o seguinte código:

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

Essa é uma função para tratamento do evento, no entanto ela está vazia. Vamos adicionar o código necessário para a inicialização:

```
procedure TFormMain.FormCreate(Sender: TObject);
begin
  kit := Tkit.Create;
  ...
```

Essa linha de código cria a instância da classe **TKit** e armazena na variável privada "kit" que criamos no início desse tutorial. Vamos utilizar essa variável "kit" para todo o controle do Kit Didático de Robótica nesse programa. Agora que já temos uma instância da classe **TKit** criada podemos utilizar os métodos que ela fornece. Um desses métodos auxilia na identificação das portas seriais instaladas no sistema operacional. Sua declaração é a seguinte:

```
Procedure GetInstalledSerialPorts(list : TStrings);
```

O método "GetInstalledSerialPorts" recebe como parâmetro uma variável do tipo Tstrings, que é uma lista de Strings, e preenche essa variável com a lista de portas seriais instaladas no sistema.

Vamos utilizar esse método para que o ComboBox mostre todas as portas seriais instaladas no sistema. A propriedade "Items" do ComboBox deve conter os ítens que serão apresentados por esse componente. Essa propriedade é do tipo TStrings e dessa forma podemos passar essa propriedade diretamente para a função "GetInstalledSerialPorts" e assim teremos nos ítens do ComboBox as portas seriais que estão instaladas no sistema. Isso pode ser feito com a seguinte linha de código:

```
kit.GetInstalledSerialPorts(ComboBoxSerial.Items);
```

Antes de enviar um comando de controle para o Kit Didático de Robótica é necessário iniciar a comunicação em uma porta serial. Isso é feito com o método "OpenCommunication" da classe de controle. A declaração desse método é a seguinte:

```
Procedure OpenCommunication(port : String);
```

Esse método recebe como parâmetro uma string com uma porta serial onde o Kit que será controlado deve estar.

Vamos fazer com que assim que o programa abra, ele tente se conectar com algum Kit na primeira porta serial encontrada no sistema. Para isso verificamos se alguma porta serial foi encontrada testando se existe algum ítem no ComboBox, se existir fazemos com que o primeiro ítem seja selecionado no ComboBox e em seguida chamamos o método "OpenCommunication" para tentar iniciar a comunicação. Isso é feito com o seguinte código:

```
// Verifica se existe algum ítem no ComboBox
if (ComboBoxSerial.Items.Count>0) then
begin
  // Seleciona o primeiro ítem
  ComboBoxSerial.ItemIndex := 0;
```

```
  // Tenta se conectar com um Kit
  kit.OpenCommunication(ComboBoxSerial.Text);
end;
```

Dessa forma já temos o código de inicialização pronto. No entanto, para evitar que mensagens de erro por exceções sejam enviadas porque não foi possível abrir a porta serial na inicialização, vamos colocar um bloco **try-except** e ignorar qualquer erro nessa inicialização. Poderíamos também enviar uma mensagem personalizada de erro, no entanto, veremos esse tipo de ação apenas em outros tutoriais. A função de inicialização completa ficaria da seguinte forma:

```
procedure TFormMain.FormCreate(Sender: TObject);
begin
  // Cria instância de TKit
  kit := TKit.Create;

  // Recupera todas a portas seriais instaladas no sistema
  kit.GetInstalledSerialPorts(ComboBoxSerial.Items);

  try
    // Verifica se existe algum ítem no ComboBox
    if (ComboBoxSerial.Items.Count>0) then
    begin
      // Se existe, seleciona o primeiro ítem
      ComboBoxSerial.ItemIndex := 0;

      // Tenta se conectar com um Kit
      kit.OpenCommunication(ComboBoxSerial.Text);
    end;
  except
  end;
end;
```

Já temos o código de inicialização, agora vamos criar o código de finalização. Novamente no **Object Inspector** selecionamos a aba de eventos e nela damos um duplo clique no evento OnDestroy. Com isso o Delphi cria o código para tratamento desse evento que ocorre quando o programa é fechado.

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

A única coisa que temos que fazer aqui é fechar a comunicação. Isso é feito com a seguinte linha de código.

```
kit.CloseCommunication;
```

O método de tratamento do evento OnDestroy fica assim.

```
procedure TFormMain.FormDestroy(Sender: TObject);
begin
  // Fecha a comunicação
  kit.CloseCommunication;
end;
```

Nesse momento podemos testar nosso programa. Ele já deve listar todas a portas seriais instaladas no sistema operacional.

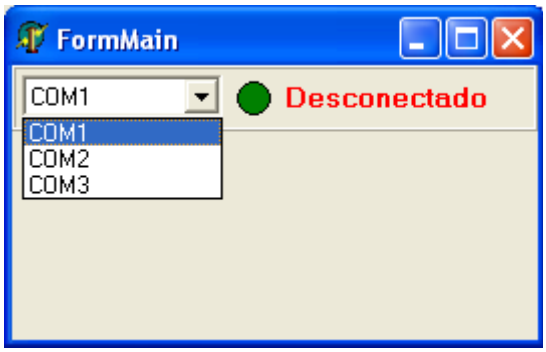

Figura 21: *Programa listando as portas seriais no sistema operacional.*

Vamos agora implementar o código para que o programa inicie a comunicação através da porta serial assim que ela for selecionada no ComboBox. Para isso utilizaremos o evento OnChange do componente ComboBox.

Selecionamos o componente ComboBox e no **Object Inspector** selecionamos a aba de eventos. Nessa aba damos duplo clique no evento OnChange. Outra alternativa é dar um duplo clique sobre o componente que automaticamente o Delphi cria o método de tratamento do evento OnChange. O seguinte código é criado:

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

Vamos adicionar o código para iniciar a comunicação assim que uma nova porta serial é selecionada. O código do evento OnChange do ComboBox ficaria da sequinte forma.

```
procedure TForm1.ComboBoxSerialChange(Sender: TObject);
begin
  // Fecha a comunicação caso essa já estivesse aberta
  kit.CloseCommunication();

  // Inicia comunicação com serial selecionada no ComboBox
  kit.OpenCommunication(ComboBoxSerial.Text);
end;
```

O último passo para finalizar nosso programa é verificar se existe algum Kit respondendo. Para isso vamos utilizar um método da classe de controle que verifica a presença de um Kit conectado e respondendo em uma porta serial. A declaração desse método é a seguinte:

```
Function  IsConnected() : Boolean;
```

Ele retorna um valor "verdadeiro" caso exista um Kit conectado à porta serial e respondendo. Se não houver nenhum, esse método retorna uma valor "falso".

Para verificar continuamente a presença de algum Kit vamos utilizar um componente Timer. Esse componente executa um método em intervalos pré-determinados. Com isso podemos identificar a conexão de um Kit a qualquer momento, uma vez que, de tempos em tempos, estaremos verificando a presença de um Kit.

O componente Timer encontra-se na aba "System" da barra de componentes. A seguir uma imagem dessa aba.

Figura 22: *Aba "System" da barra de componentes.*

O componente Timer possui o seguinte ícone:

Figura 23: *Ícone do componente Timer.*

Temos que adicionar um componente desses no nosso Form principal e modificar as seguintes propriedades dele:

Name	=	TimerCheck
Interval	=	250

Com isso nosso Timer irá executar o evento OnTimer a cada 250 milisegundos, ou seja, quatro vezes por segundo. Temos que implementar esse evento para verificar a presença de um Kit. Podemos fazer isso selecionando o componente Timer e no **Object Inspector** selecionar a aba de eventos e dar um duplo clique no evento OnTimer. Outra maneira mais simples é dar um duplo clique sobre o componente Timer diretamente e o Delphi irá implementar o método de tratamento do evento OnTimer, pois esse é o evento padrão desse componente.

Com isso, já com o código para o método de tratamento do evento OnTimer, vamos adicionar nosso código, seguindo o modelo abaixo.

```
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
  else
  begin
    // "Apaga" LED. Muda cor de fundo do Shape
    ShapeLED.Brush.Color := clGreen;
```

```
        // Muda texto para indicar erro de conexão
        LabelConnected.Caption := 'Desconectado';

        // Muda cor do texto para vermelho
        LabelConnected.Font.Color := clRed;
    end;
end;
```

Essa rotina verifica se existe um Kit conectado e modifica as propriedades do Shape e do Label da barra superior de acordo com o estado da conexão. Dessa forma podemos informar ao usuário o estado da conexão de forma gráfica.

Existe um detalhe que temos que tratar nesse momento. Quando chamamos o método "IsConnected" e não existe um Kit conectado e respondendo na porta serial selecionada, a biblioteca de controle do Kit emite uma exceção, que é tratada internamente no método e deveria passar despercebida durante a execução do programa.

No entanto, em tempo de desenvolvimento, o Borland Delphi indica que essa exceção ocorreu e interrompe a execução do programa. Como nossa verificação é feita dentro de um Timer que é chamado algumas vezes por segundo, então essa interrupção na execução do programa irá impedir qualquer teste de execução dentro do Delphi se não existir um Kit conectado e respondendo na porta serial selecionada.

Para que possamos continuar desenvolvendo e debugando nossos programas normalmente, temos duas opções. A primeira é desabilitar o Timer durante o desenvolvimento. Dessa forma não teremos mais uma resposta visual indicando o estado da conexão, no entanto, o resto do programa funcionará normalmente. Assim que o programa estiver finalizado podemos habilitar novamente o Timer, compilar o programa e a partir dai utilizar o executável que foi gerado. Para desabilitar o Timer temos que selecionar esse componente e modificar sua propriedade "Enabled" de true para false.

Outra opção é evitar que o Delphi interrompa a execução do programa quando ocorre uma exceção, dessa forma podemos manter a resposta visual do estado da conexão. Podemos fazer isso indo na opção de menu **Tools – Debugger Options...** , será apresentada uma janela de configurações. Entre na aba "Languages Exceptions" e nessa aba desabilite a opção "Stop on Delphi Exceptions". Com isso o Delphi não irá mais parar nas exceções. A seguir uma figura da tela onde essa opção deve ser desabilitada.

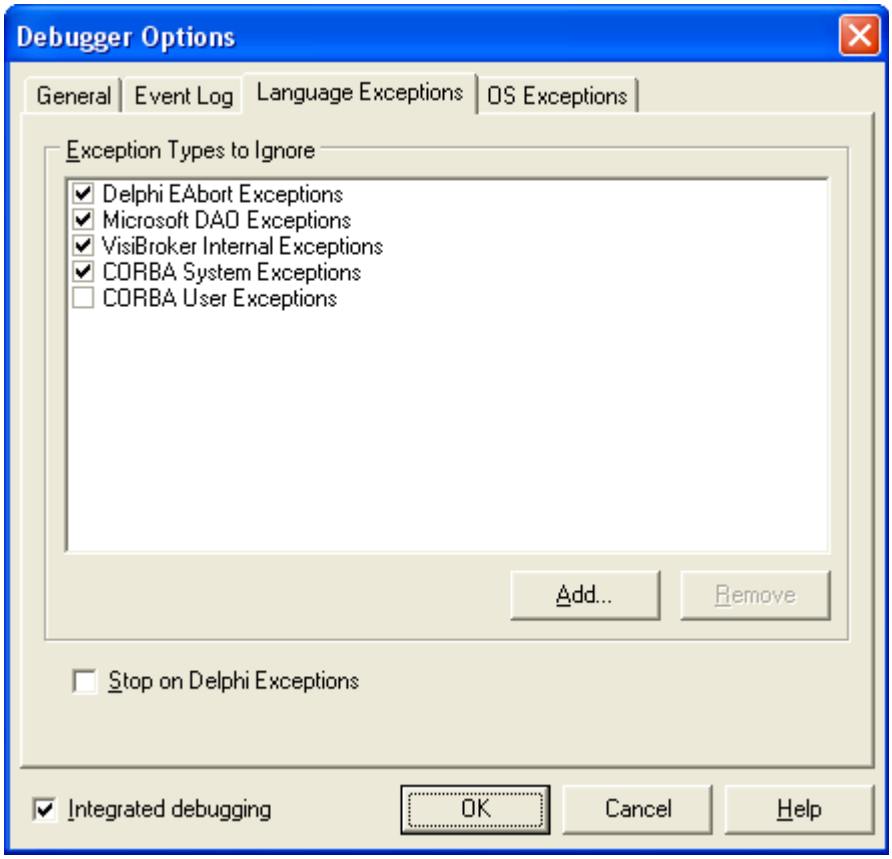

Figura 24: *Configuração para que o Delphi não pare a execução de um programa em tempo de desenvolvimento se uma exceção ocorrer e for tratada internamente.*

A seguir uma imagem com o programa final e com os dois estados da conexão. No primeiro a aparência sem conexão e no segundo com conexão.

Figura 25: *Programa base indicando o estado da conexão. Do lado esquerdo aparência quando não há conexão e do lado direito com uma conexão normal.*

4 – Dicas

A seguir são dadas algumas dicas para facilitar o desenvolvimento de softwares de controle do Kit.

A primeira dica é sobre como organizar a pasta do projeto. É interessante que os arquivos de código fonte sejam separados de arquivos executáveis e de arquivos intermediários criados pelo compilador. Para isso é interessante criar a seguinte árvore de diretório:

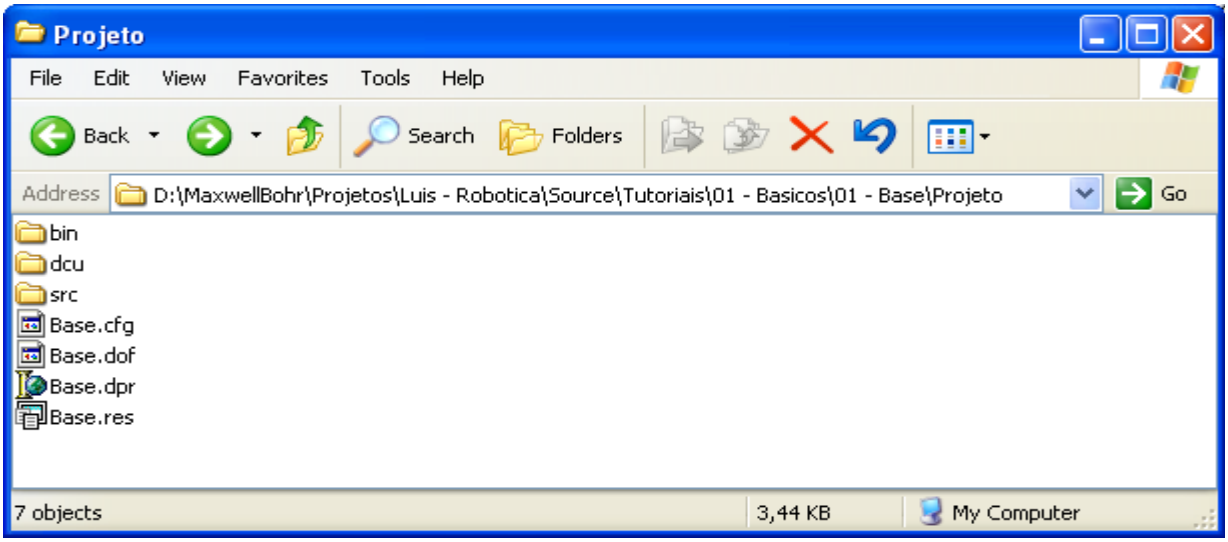

Figura 26: *Estrutura do diretório do projeto.*

Nessa estrutura de arquivos vamos fazer com que os arquivos executáveis sejam armazenados no diretório “bin”, os arquivos intermediários da compilação no diretório “dcu” e os arquivos fonte no diretório “src”. Os arquivos do projeto ficam soltos no diretório.

Todos os arquivos com código fonte ou arquivos de suporte a formulários serão salvos na pasta “src”. Essa pasta contém os seguintes arquivos no nosso projeto.

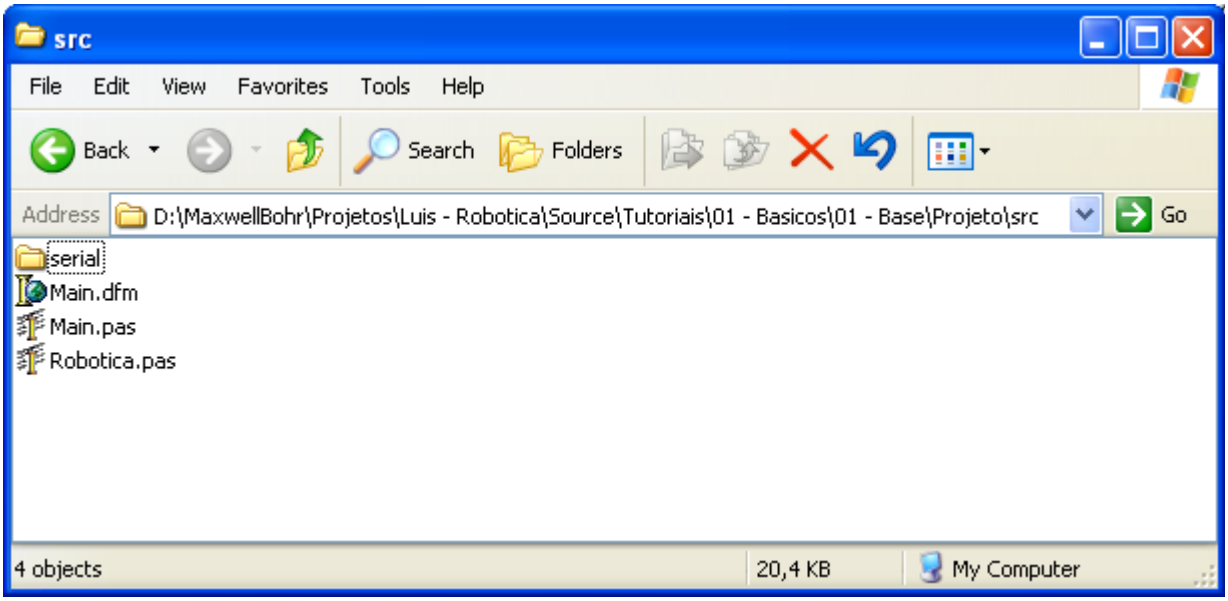

Figura 27: *Arquivos da pasra "src".*

Temos também que configurar nas opções de projeto do Borland Delphi para que ele utilize a pasta “bin” para armazenar o executável e a pasta “dcu” para arquivos intermediários. Para isso, com nosso projeto aberto, vamos na opção de menu **Project – Options...** e será apresentada a seguinte janela:

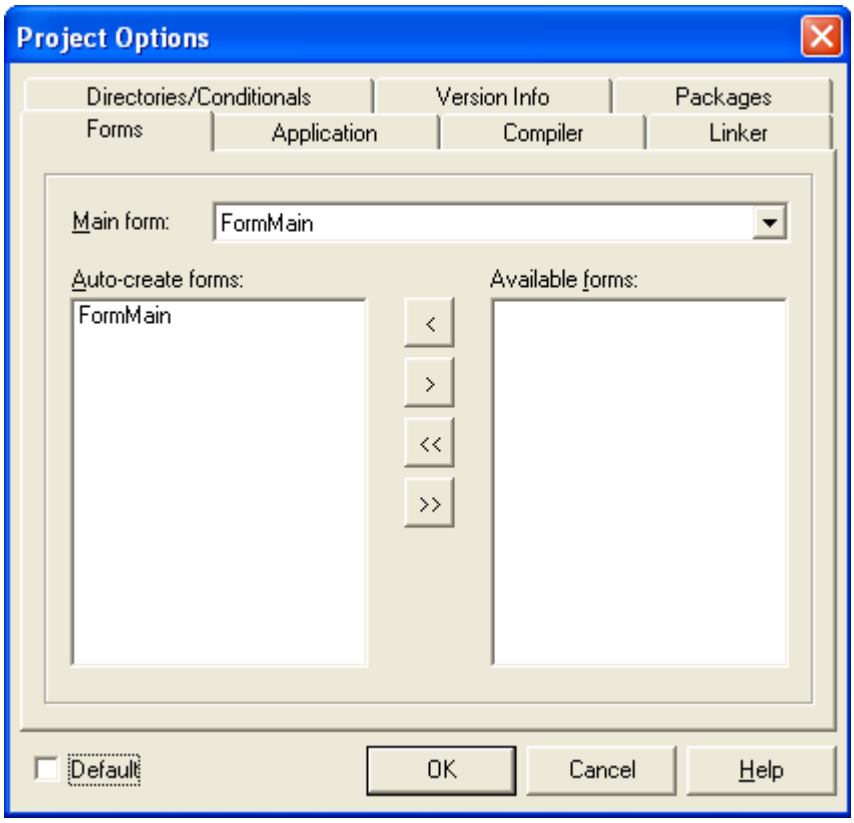

Figura 28: *Janela de configurações do projeto.*

Para configurar os diretórios de saída vamos na aba “Directories/Conditionals”. Será

apresentada a seguinte janela:

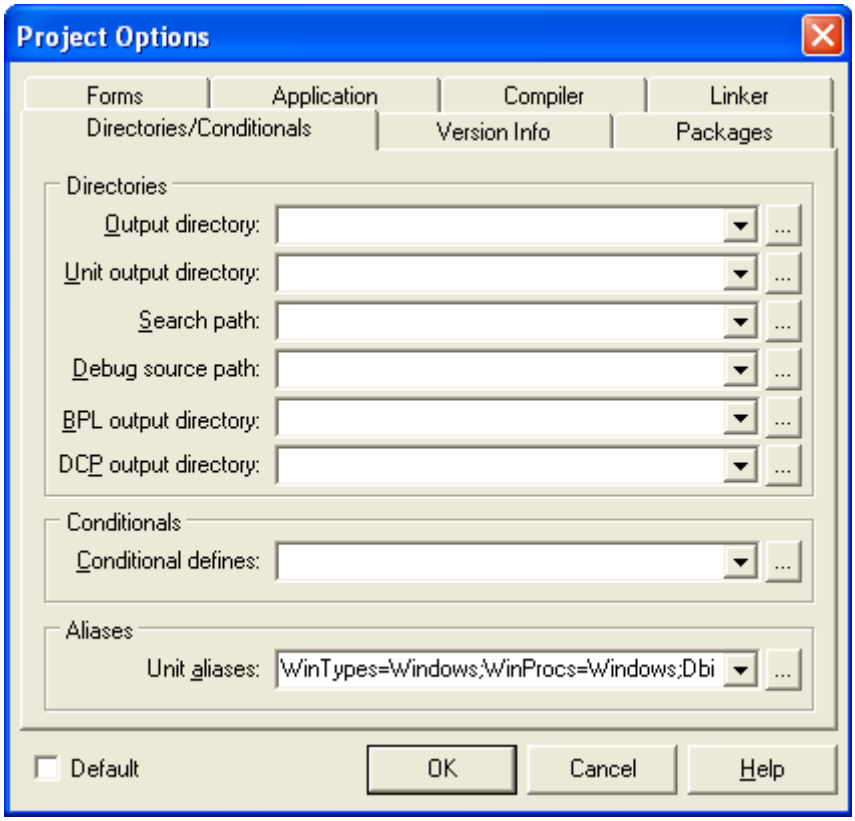

Figura 29: *Configuração de diretórios.*

Devemos alterar as seguintes opções:

Output directory	=	bin
Unit output directory	=	dcu

A primeira opção indica que o arquivo executável deve ser armazenado na pasta "bin" e a segunda opção indica que os arquivos intermediários devem ser armazenados na pasta "dcu". Com isso quando compilarmos o projeto os arquivos executáveis e intermediários serão gravados automaticamente em diretórios diferentes do diretório dos arquivos de código fonte.

Outra dica interessante é configurar o projeto para que ele crie um arquivo executável com todas as bibliotecas necessárias embutidas. Com isso não é necessário que arquivos de biblioteca sejam distribuidos junto com o arquivo executável para que esse funcione. A falta de arquivos de biblioteca é um problema muito comum e com essas configurações que vamos fazer podemos evitar problemas.

Na janela de configurações do projeto vá na aba "Packages". Na parte inferior dessa janela existe uma opção denominada "Build with runtime packages". Ela fica selecionada por padrão, descelecione essa opção. Dessa forma as bibliotecas serão incluídas no arquivo executável.

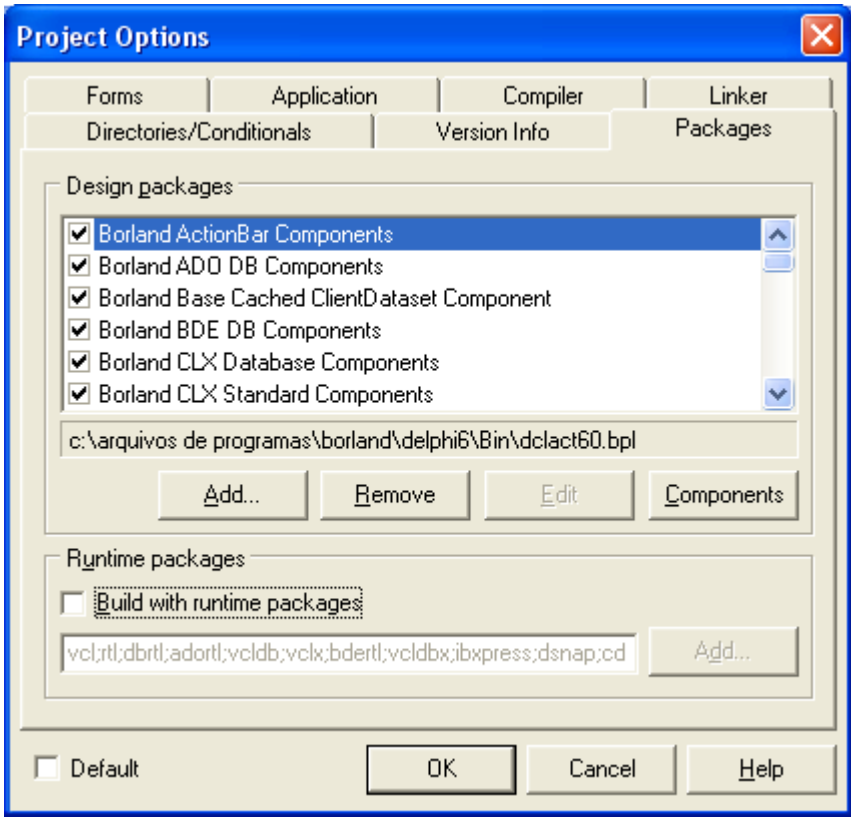

Figura 30: *A opção "Build with runtime packages" deve ser desabilitada para evitar a necessidade de distribuição de arquivos de biblioteca quando o programa for executado em outras máquinas.*

5 – Conclusão

Nesse tutorial vimos como criar um programa base para controlar o Kit Didático de Robótica. A partir do programa criado nesse tutorial podem ser criados outros programas com maior facilidade, pois toda a estrutura de base já está implementada. Todos os outros tutoriais vão utilizar o programa criado nesse tutorial como base. Se esse tutorial for bem entendido não haverá problemas para o entendimento dos outros tutoriais e da programação para o Kit.