TUTORIAL
INFORMAÇÕES SOBRE O KIT

Autor: Tiago Lone
Nível: Básico
Criação: 08/12/2005
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 como obter informações sobre o Kit Didático de Robótica. Com esse tipo de informação podemos tomar decisões com maior segurança nos programas que criamos, uma vez que, saberemos exatamente a configuração do dispositivo com que estamos interagindo. Para demonstrar na prática o que vamos aprender no tutorial, será criado um programa que obtém e apresenta todas as informações de um Kit Didático de Robótica.

2 – Material

Para esse tutorial será necessário o *Módulo Principal* de um KDR5000 conectado em uma porta serial do computador. No entanto, o uso alternado do *Módulo de Motores e Displays* e o *Módulo de Entradas, Saídas e Servo-Motores* é interessante pois existem algumas informações que dependem desses módulos. Para a criação do programa será necessário o Borland Delphi 6.

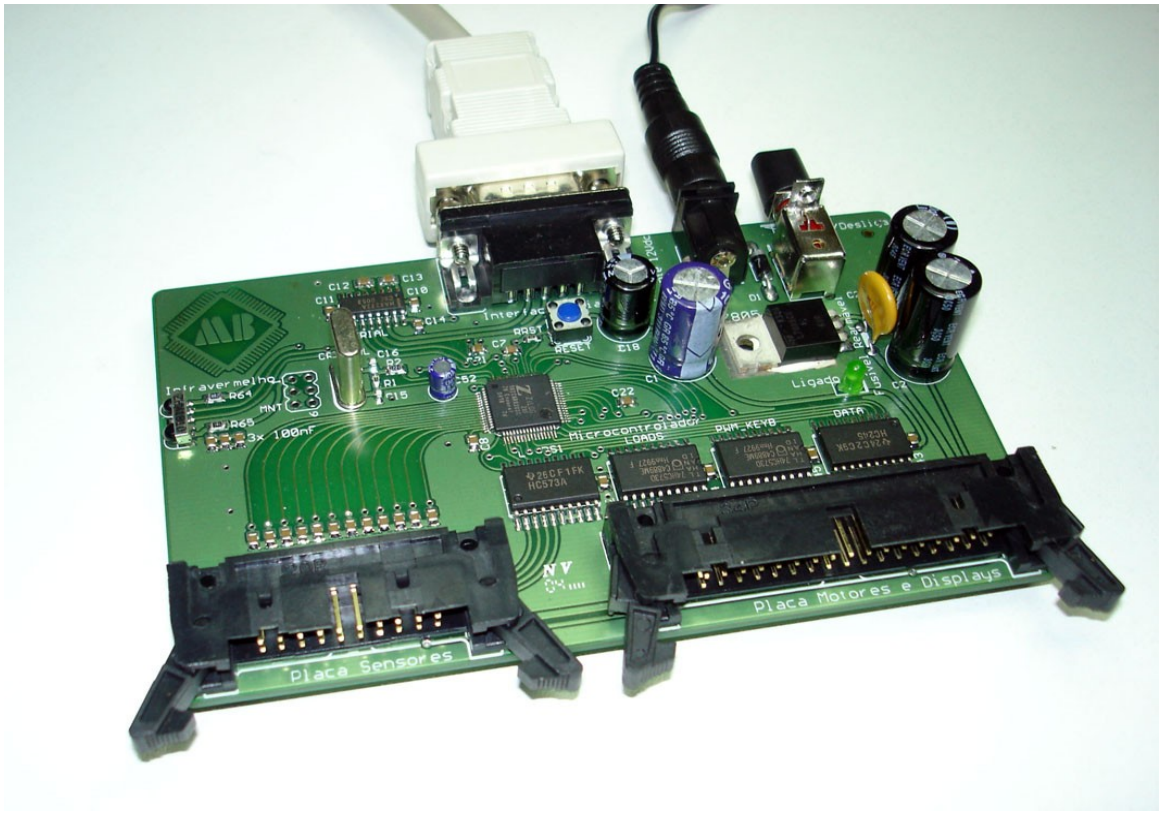

Figura 1: *Módulo Principal do Kit Didático de Robótica.*

3 – Projeto

Para aprender o que devemos fazer para obter informações sobre o Kit com que estamos nos comunicando, vamos criar um projeto que irá apresentar todas as informações do Kit em uma janela. A seguir, uma imagem da aparência final do programas que iremos criar.

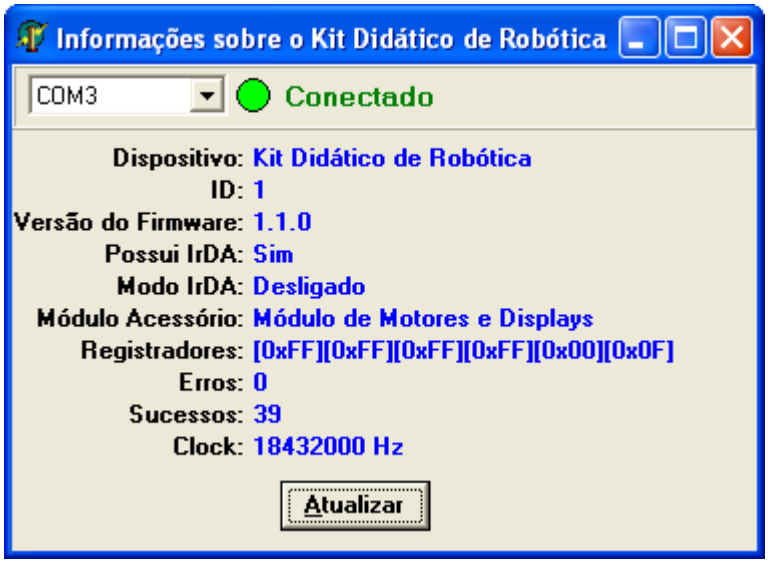

Figura 2: *Aparência final do programa que será criado nesse tutorial.*

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

A interface do programa é bem simples, composta apenas por vários componentes gráficos do tipo Label, que apresentam as informações do Kit, e um componente Button, que é um botão, utilizado para atualizar as informações. Esses dois componentes podem ser encontrados na aba de componentes “Standard”.

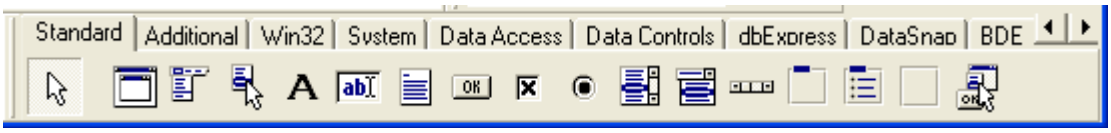

Figura 3: *Aba "Standard" da Barra de componente.*

O componente Label possui o seguinte ícone.

Figura 4: *Ícone do componente Label.*

O componente Button possui o seguinte ícone.

Figura 5: *Ícone do componente Button.*

Temos que adicionar à interface que foi criada no tutorial Base duas colunas de componentes Label, uma com o nome da informação que será apresentada e outra com a informação propriamente dita. Vamos apresentar dez informações diferentes, logo serão necessários dez Labels por coluna. A figura a seguir mostra como o Form principal irá se parecer após a adição das duas colunas de Labels.

Figura 6: *Adição de duas colunas de componentes Labels.*

Após adicionar os Labels, conforme a disposição apresentada na figura anterior, vamos modificar algumas de suas propriedades. Modificaremos primeiro as propriedades dos Labels da coluna da esquerda. Cada um desses Labels possui um texto com o nome de uma informação. Esse texto será apresentado em negrito. É interessante também modificar o nome do componente para um nome mais descritivo. A seguir, as propriedades dos dois primeiros Labels. Os outros seguem o mesmo padrão, bastando apenas adaptá-los para apresentarem a infomação correta.

Name	=	LabelDispositivo
Caption	=	Dispositivo:
Font/Style/fsBold	=	true
Alignment	=	taRightJustify

Propriedades do segundo Label da coluna da esquerda.

Name	=	LabelID
Caption	=	ID:
Font/Style/fsBold	=	true
Alignment	=	taRightJustify

Os nome das dez informações são “Dispositivo”, “ID”, “Versão do Firmware”, “Possui IrDA”, “Modo IrDA”, “Módulo Acessório”, “Registradores”, “Sucessos”, “Erros” e “Clock”. Os Labels da coluna da esquerda deverão apresentar esses textos.

Após modificar as propriedades dos Labels da coluna da esquerda, temos que fazer os últimos ajustes na posição desses componentes para que eles formem uma coluna alinhada, como demonstrado na figura da interface pronta.

Com a coluna da esquerda finalizada, temos que criar a coluna de Labels da direita. Essa coluna irá apresentar as informações coletadas do Kit. Essas informações serão atualizadas somente quando o programa estiver em execução, por isso, podemos deixar o texto deles com um valor qualquer, sendo que no caso vamos utilizar um traço. Com isso as propriedades desses componentes serão muito parecidas. A seguir, as propriedades do primeiro Label da coluna da direita. Os outros Labels dessa coluna seguirão o mesmo padrão.

Name	=	LabelInfoDispositivo
Caption	=	-
Brush/Color	=	clBlue
Font/Style/fsBold	=	true

Com as duas colunas de Labels feitas, falta apenas um ítem na nossa interface gráfica, o botão de atualização. Vamos adiciona-lo abaixo das colunas e modificar as seguintes propriedades.

Name	=	ButtonAtualizar
Caption	=	Atualizar
Font/Style		

Com isso, a

Agora que nificam as informações apresentadas por ela. A presentadas.

Figura 7: *Form com a aparência final. Duas colunas de Labels e um botão de atualização.*

Informação	*Descrição*
Dispositivo	Nome descritivo do modelo do Kit que estamos nos comunicando.
ID	Número de identificação do modelo do Kit.
Versão do Firmware	Versão do programa (Firmware) que roda no microcontrolador do Kit. Possui versão, sub-versão e release.
Possui IrDA	Indica se o Kit possui porta de comunicação infravermelho.
Modo IrDA	Indica se o modo de envio de comandos através da porta de comunicação infravermelho esta ligado.
Módulo Acessório	Indica qual módulo está conectado ao conector para cabo flat de 34 vias do *Módulo Principal*.
Registradores	Estado dos registradores de controle internos do Kit.
Sucessos	Indica o número de comandos enviados e executados com sucesso.
Erros	Indica o número de comandos que falharam, ou por erro na execução ou por falha de comunicação.
Clock	Indica qual é o clock de operação do microcontrolador do Kit em Hertz.

Essas são as principais informações que podemos obter sobre um Kit. Vamos agora adicionar o código que irá obter esses dados do Kit e com isso, atualizar o texto dos Labels.

O primeiro passo para isso é conhecer os métodos da biblioteca de controle que nos oferece essa funcionalidade. São apenas dois métodos, um que utilizamos para obter o nome descritivo do modelo do Kit e que possui a seguinte declaração.

```
Function  DeviceName() : String;
```

Esse método é bem simples, ele retorna uma string com o nome descritivo do modelo do Kit com que estamos nos comunicando. Vamos criar um manipulador para o evento OnClick do nosso botão de atualização. Para fazer isso podemos selecionar o componente Button, 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 adicionar o código para atualizar a informação sobre o nome do Kit dentro desse manipulador, dessa forma ao pressionar o botão o nome do dispositivo já será atualizado.

```
Procedure TFormMain.ButtonAtualizarClick(Sender: TObject);
begin
  // Atualiza nome do dispositivo
  LabelInfoDispositivo.Caption := Kit.DeviceName;
end;
```

Adicionando esse código já podemos testar o programa. Para isso vamos no menu **Run – Run** ou pressionamos F9. Se não houver nenhum erro o programa será compilado e executado. Com um Kit conectado em alguma porta serial podemos testar se o programa está funcionando, selecionamos a porta serial correta e pressionamos o botão "Atualizar". Com isso o texto em frente da informação "Dispositivo" será atualizado e deverá se parecer com o seguinte.

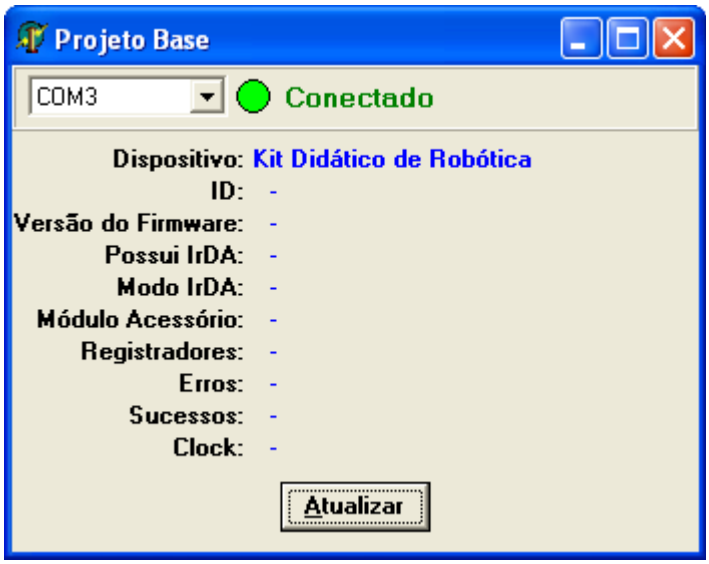

Figura 8: *Teste do programa. Apresentação do nome descritivo do modelo do Kit.*

Para obter as outras informações será necessário o uso do segundo método de obtenção de informações sobre o Kit. Usaremos o método DeviceStatus. A seguir a declaraçã odesse método.

```
Procedure DeviceStatus(var status : DynByteArray);
```

Esse método é um pouco mais complexo do que o DeviceName. Ele retorna um array de bytes com várias informações ao mesmo tempo fazendo uso do parâmetro "status", que é um parâmetro passado por referência, o que possibilita que esse seja utilizado para retornar um valor.

O parâmetro "status" é do tipo DynByteArray, que é uma classe para arrays dinâmicos de bytes. Podemos trabalhar com ela como se fosse um array normal, no entanto, essa classe fornece muito mais funcionalidades que um array normal. Aqui vamos utilizar apenas as funcionalidades básicas. Teremos apenas que ler os valores que serão gravados nesse array dinâmico como se fosse um array simples.

Apenas para ressaltar, apesar de "status" ser um parâmetro, esse pode ser utilizado para

retornar um valor pois é passado por referência, observe a palavra chave "**var**" na frente do parâmetro "status" na declaração do método. Quando parâmetros são passados por referência, uma modificação no parâmetro, feita internamente no método, afetará a variável original. Dessa forma podemos passar valores de dentro do método para fora. Esse conceito ficará mais claro em seguida ao criarmos o código do nosso programa. Para mais detalhes sobre parâmetros passado por referência procure alguma documentação sobre programação em Delphi.

Voltando ao nosso retorno de um array de bytes com as informações do Kit, a seguir está detalhado como estão codificadas as informações nesse array na ordem em que aparecem. Em seguida veremos o que é necessário fazer para decodificar esses valores.

Informação	*Número de Bytes*	*Descrição*
ID	2	Número de identificação do modelo do Kit.
Versão do Firmware	4	Versão do programa que roda no microcontrolador do Kit. Possui versão, sub-versão e release.
Possui IrDA	1	Indica se o Kit possui porta de comunicação infravermelho.
Modo IrDA	1	Indica se o modo de envio de comandos através da porta de comunicação infravermelho esta ligado.
Módulo Acessório	1	Indica qual módulo está conectado ao conector para cabo flat de 34 vias do *Módulo Principal*.
Registradores	6	Estado dos registradores de controle internos do Kit.
Sucessos	2	Indica o número de comandos enviados e executados com sucesso.
Erros	2	Indica o número de comandos que falharam, ou por erro na execução ou por falha de comunicação.
Clock	4	Indica qual é o clock de operação do microcontrolador do Kit.
Reservados	5	Reservados ou uso interno.

Agora que já sabemos como é o array de bytes com as informações sobre o Kit, vamos implementar o restante do código do nosso programa.

```
Procedure TFormMain.ButtonAtualizarClick(Sender: Tobject);
var
  // Array passado como parâmetro ao método DeviceStatus
  info : DynByteArray;

begin
  // Atualiza nome do dispositivo
  LabelInfoDispositivo.Caption := Kit.DeviceName;
```

```
    // Obtem as informações do Kit e armazena em info
    Kit.DeviceStatus(info);
end;
```

Com essas linhas adicionais de código criamos um objeto do tipo DynByteArray chamado "info" e chamamos o método DeviceStatus para retornar nesse objeto os bytes com as informações que desejamos. Agora só nos falta atualizar o texto dos Labels. É isso que vamos vazer em seguida.

```
Procedure TFormMain.ButtonAtualizarClick(Sender: Tobject);
var
    // Array passado como parâmetro do método DeviceStatus
    info: DynByteArray;

begin
    // Atualiza nome do dispositivo
    LabelInfoDispositivo.Caption := Kit.DeviceName;

    // Obtem as informações do Kit e armazena em info
    Kit.DeviceStatus(info);

    // Atualiza os Labels
    // ID
    LabelInfoID.Caption := IntToStr((info[0] SHL 8)
                                        OR info[1]);

    ...

end;
```

Como a informação de ID é um valor formado por dois bytes, precisamos unir seus dois bytes para formar um valor de 16 bits. O primeiro byte é a parte mais significativa do valor e o segundo a menos significativa. Para formar o valor temos que deslocar o byte mais significativo 8 posições para a esquerda e em seguida inserir o byte menos significativo no valor. Para fazer isso deslocamos o byte mais significativo utilizando o operador de deslocamento para a esquerda do Delphi, o "SHL", e em seguida inserimos a parte menos significativa do valor utilizando o operador "OR". A seguir essa parte do código isolada.

```
(info[0] SHL 8) OR info[1]
```

Nessa parte do código, acessamos o primeiro byte do array, deslocamos ele 8 posições para a esquerda e em seguida inserimos a parte menos significativa do valor, que é o segundo byte, utilizando o operador binário "OR".

Com isso temos o valor de 16 bits do ID. Temos que converter esse valor para uma "String" antes de apresentar a informação no Label. Para isso utilizamos o método IntToStr do Delphi para conversão de valores "Integer" para "String" e em seguida atribuimos esse valor à propriedade "Caption" do componente LabelIndoID.

Vamos agora recuperar a versão do firmware, para isso adicionamos o seguinte código ao nosso programa.

```
  ...
  // Atualiza os Labels
  // ID
  LabelInfoID.Caption := IntToStr((info[0] SHL 8)
                                          OR info[1]);

  // Versão do Firmware
  LabelInfoVersaoFirmware.Caption := IntToStr(info[2]) +
                          '.' + IntToStr(info[3]) + '.' +
                    IntToStr((info[4] SHL 8) AND info[5]);
  ...
end;
```

A informação de versão do firmware é formada por 4 bytes. O primeiro indica a versão, o segundo a sub-versão e os dois últimos formam um valor de 16 bits que indica o release. Para formar esse valor de 16 bits de release, fazemos as mesmas operações que fizemos para extrair a informação de ID.

Utilizamos o método IntToStr para converter os valores numéricos para "Strings" e formamos uma string de versão, sub-versão e release completa, fazendo uso do operador de soma para concatenar as "Strings". Nesse momento podemos testar nosso programa para garantir que está tudo certo até o momento. Sua aparência será a seguinte.

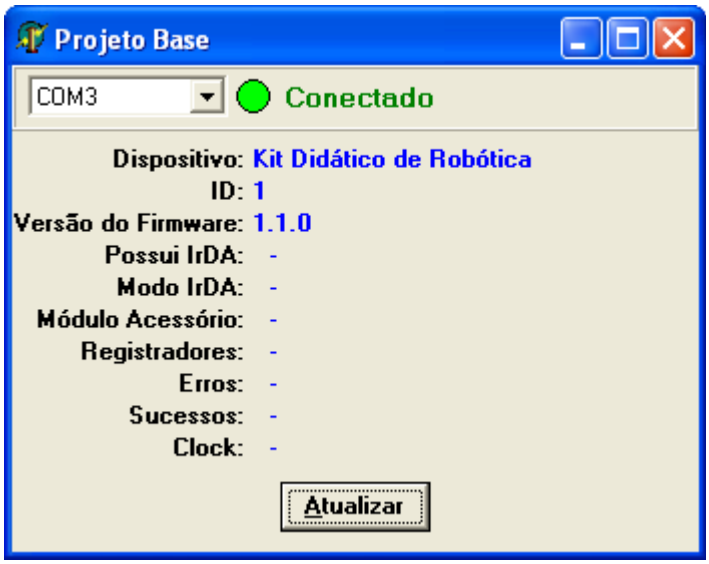

Figura 9: *Programa apresentando nome descritivo do modelo, número de identificação do modelo e versão do firmware.*

Vamos adicionar mais algumas linhas.

```
...

// Versão do Firmware
LabelInfoVersaoFirmware.Caption := IntToStr(info[2]) +
                        '.' + IntToStr(info[3]) + '.' +
                IntToStr((info[4] SHL 8) AND info[5]);

// Possui IrDA
if(info[6] = 0) then
  LabelInfoIrDA.Caption := 'Não'
else
  LabelInfoIrDA.Caption := 'Sim';

// Modo IrDA
if(info[7] = 0) then
  LabelInfoIrDAOn.Caption := 'Desligado'
else
  LabelInfoIrDAOn.Caption := 'Ligado';
```

```
  ...
end;
```

Esse novo bloco de código verifica se o Kit possui uma porta infravermelho. Se o byte dessa informação for "0", então o Kit não possui uma porta infravermelho, caso contrário ele possui. A outra informação que verificamos é se o modo de envio de comandos pela porta infravermelho está ativado. Se o valor dessa informação for "0", o modo está desligado, caso contrário ele está ligado.

Nossa próxima informação é o módulo que está conectado ao Kit. O código para verificar isso é o seguinte.

```
  ...

  // Modo IrDA
  if(info[7] = 0) then
    LabelInfoIrDAOn.Caption := 'Desligado'
  else
    LabelInfoIrDAOn.Caption := 'Ligado';

  // Módulo Acessório
  if(info[8] = 1) then
    LabelInfoModuloAcessorio.Caption := 'Módulo de Motores
                                          e Displays'
  else if(info[8] = 2) then
    LabelInfoModuloAcessorio.Caption := 'Módulo de
                        Entradas, Saídas e Servo-Motores'
  else
    LabelInfoModuloAcessorio.Caption := 'Desconhecido';

  ...
end;
```

Se o valor da informação sobre o módulo conectado for igual a "1" então não há nenhum módulo conectado ou o módulo é o *Módulo de Motores e Displays*, nesse exemplo, caso o valor seja "1" vamos sempre apresentar que o *Módulo de Motores e Displays* está conectado. Se o valor da informação for igual a "2", então o *Módulo de Entradas, Saídas e Servo-Motores* que está conectado. Qualquer outro valor é de algum módulo desconhecido e não deveria ocorrer.

A próxima informação que vamos apresentar são os registradores de controle interno do

Kit. Vejamos a seguir.

```
...

// Módulo Acessório
if(info[8] = 1) then
  LabelInfoModuloAcessorio.Caption := 'Módulo de Motores
                                       e Displays'
else if(info[8] = 2) then
  LabelInfoModuloAcessorio.Caption := 'Módulo de
                      Entradas, Saídas e Servo-Motores'
else
  LabelInfoModuloAcessorio.Caption := 'Desconhecido';

// Registradores
LabelInfoRegistradores.Caption :=
          '[0x' + IntToHex(info[9], 2) + ']' +
          '[0x' + IntToHex(info[10], 2) + ']' +
          '[0x' + IntToHex(info[11], 2) + ']' +
          '[0x' + IntToHex(info[12], 2) + ']' +
          '[0x' + IntToHex(info[13], 2) + ']' +
          '[0x' + IntToHex(info[14], 2) + ']';

...
end;
```

Aqui apenas recuperamos os valores dos registradores e utilizamos o método IntToHex para transformar os valores em uma string com a representação hexadecimal dos valores. Para maiores informações sobre esse método veja a ajuda do proprio Delphi.

Temos mais dois valores de 16 bits para recuperar, o número de comandos que falharam e o número dos que foram executados com sucesso. Vamos utilizar as mesmas operações que utilizamos anteriormente para transformar os dois bytes dessas informações em um valor de 16 bits. Fazemos isso com o seguinte código.

```
...

// Registradores
```

```
    LabelInfoRegistradores.Caption :=
              '[0x' + IntToHex(info[9], 2) + ']' +
              '[0x' + IntToHex(info[10], 2) + ']' +
              '[0x' + IntToHex(info[11], 2) + ']' +
              '[0x' + IntToHex(info[12], 2) + ']' +
              '[0x' + IntToHex(info[13], 2) + ']' +
              '[0x' + IntToHex(info[14], 2) + ']';

    // Erros
    LabelInfoErros.Caption := IntToStr((info[15] SHL 8)
                                       OR info[16]);

    // Sucessos
    LabelInfoSucessos.Caption := IntToStr((info[17] SHL 8)
                                       OR info[18]);
    ...
end;
```

Por fim vamos apresentar a informação sobre o clock de operação do microcontrolador do Kit. Isso é feito da seguinte maneira.

```
    ...

    // Erros
    LabelInfoErros.Caption := IntToStr((info[15] SHL 8)
                                       OR info[16]);

    // Sucessos
    LabelInfoSucessos.Caption := IntToStr((info[17] SHL 8)
                                       OR info[18]);

    // Clock
    LabelInfoClock.Caption := IntToStr((info[19] SHL 24) OR
                                  (info[20] SHL 16) OR
                                  (info[21] SHL 8) OR
```

```
                                                    info[22]) + ' Hz';
        end;
```

Aqui transformamos 4 bytes em um valor de 32 bits com a mesma lógica que utilizamos anteriormente para tranformar 2 bytes em um valor de 16 bits.

Um último detalhe que vamos modificar é a propriedade Caption do Fom principal. Como copiamos o projeto Base, essa propriedade ainda esta como " Projeto Base", vamos modificar para "Informações sobre o Kit Didático de Robótica". Com isso terminamos a implementação do código do nosso programa. A aparência final dele será a seguinte.

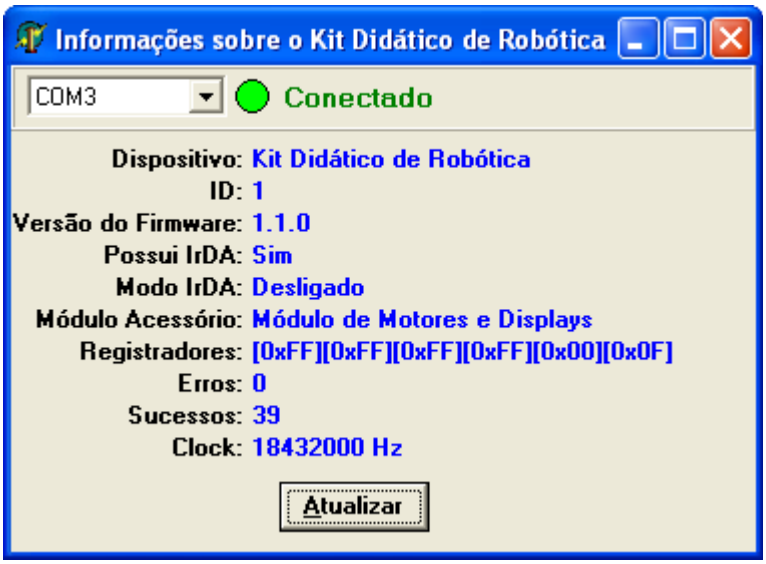

Figura 10: *Programa finalizado.*

4 – Conclusão

Nesse tutorial vimos como obter informações sobre o Kit com que estamos nos comunicando. Com isso podemos saber exatamente as características do dispositivo e tomar qualquer decisões que dependam dessas informações baseadas em informações que foram obtidas diretamente do Kit.