Comunicado 109
Técnico
ISSN 1517-4786
Julho, 2009
São Carlos, SP

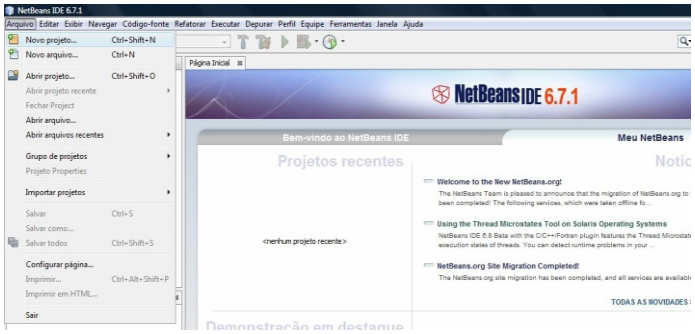

Foto: Ladislau Marcelino Rabello

Programa em linguagem JAVA para comunicação Serial

Ladislau Marcelino Rabello

Introdução

Equipamentos microprocessados ou microcomputadores vêem equipados com um recurso de comunicação que propicia a troca de dados entre dois sistemas.

Este tipo de comunicação pode tanto ser via serial, os dados são enviados via seqüencial, ou via paralela em que os dados são enviados como o próprio nome diz de forma paralela.

Para se fazer um sistema comunicar-se com outro é necessário bom conhecimento dos circuitos e padrões de comunicação e também de uma linguagem de programação. A linguagem Java vem se tornando muito comum entre programadores, por ser uma linguagem de alto nível e que foi desenvolvida para trabalhar em diferentes sistemas operacionais, tal como o Windows e o Unix, mais precisamente na sua versão mais popular o Linux.

Uma vez instalados os compiladores Java nos sistemas operacionais, o programa pode ser escrito em um editor de texto qualquer e posto para se executado em qualquer destes sistemas operacionais.

Existem várias bibliotecas desenvolvidas pelos projetistas da linguagem Java que auxiliam em muito os programadores, porém as bibliotecas para comunicação serial foram por assim dizer deixadas de lado pelos projetistas da linguagem.

A grande maioria das bibliotecas desenvolvidas para comunicação serial foi realizada por projetistas de outras instituições. Assim o objetivo deste comunicado é de discutir e apresentar uma biblioteca desenvolvida pela RxTx em linguagem java.

Definição de comunicação serial

O sistema de comunicação via dois ou mais computadores, é possível via interface serial, já inerente na maioria dos computadores. A comunicação serial é um processo de transferência de dados de um bit por vez. As comunicações seriais incluem a maioria dos dispositivos de rede, teclados, mouses, modems e terminais.

Quando se é feita uma comunicação serial cada palavra (isto é, *byte* ou caractere) de dados que é enviado ou recebido é feito a um bit por vez. Cada bit pode ser representado pelo estado lógico ligado (nível lógico 1) ou desligado(nível lógico 0).

A velocidade de envio dos dados por uma porta serial é expressa em bits-por-segundo (bps, na nomenclatura inglesa) ou em *baudot rate* (“*baud*”), que representam o número de bits (zeros ou uns) que podem ser enviados ou recebidos em um segundo. Nos computadores estas velocidades podem atingir 19.2k bps, ou mais.

Definição de RS-232

RS é uma abreviação de *Recommended Standard*, a qual relata uma padronização de uma interface comum para comunicação de dados entre equipamentos. Criada no início dos anos 60 por um comitê conhecido como *Electronic Industries Association* (EIA).

Nesta época, a comunicação de dados compreendia a troca de dados digitais entre um computador central (*mainframe*) e terminais de computador remotos, ou entre dois terminais sem o envolvimento do computador.

Os dispositivos podiam ser conectados através de linha telefônica e conseqüentemente necessitavam de um demulador em cada lado para fazer a decodificação dos sinais.

No intuito de padronizar estes sinais é que surgiu o padrão RS-232, onde ele específica as tensões, temporizações e funções de sinais, um protocolo para troca de informações e as conexões mecânicas.

[1]Engenharia Eletrônica, Dr., Pesquisador, Embrapa Instrumentação Agropecuária, C.P. 741, CEP 13560-970, São Carlos SP, rabello@cnpdia.embrapa.br

Definição de sinais

Nas tabelas 1 e 2 são definidos as conexões para um sistema DTE (*data terminal equipament*), que normalmente seria o micro computador PC, .

Tabela 1. Pinagem para um sistema DTE, conector DB-25(macho)

Pino	Descrição
1	Shield
‘2	Transmitted data
3	Received data
4	Request to send
5	Clear to send
6	DCE ready
7	Signal ground
8	Received line - signal detect
9	Reserved for testing
10	Reserved for testing
11	Unassigned
12	Sec. Received line – signal detect
13	Sec. Clear to send
14	Sec. Transmetted data
15	Transmitter signal timing (DEC source)
16	Local loopback
17	Receiver signal timing (DEC source)
18	Local loopback
19	Sec. Request to send
20	DTE ready
21	Remote loopback
22	Ring indicator
23	Data signal rate selector
24	Transmitter signal timing (DTE source)
25	Test mode

Tabela 2. Pinagem para um sistema DTE, conector DB-9(macho)

Pino	Descrição
1	Received line signal detect
2	Received data
3	Transmitted data
4	DTE ready
5	Signal ground
6	DCE ready
7	Request to send
8	Clear to send
9	Ring indicator

Nas tabelas 3 e 4 são definidos as pinagens para um sistema DCE (*data circuit-terminating equipament*), que normalmente seria o demulador.

Tabela 3. Pinagem para um sistema DCE, conector DB-25(femea)

Pino	Descrição
1	Shield
‘2	Received data
3	Transmitted data
4	Clear to send
5	Request to send
6	DCE ready
7	Signal ground
8	Received line - signal detect
9	Reserved for testing
10	Reserved for testing
11	Unassigned
12	Sec. Received line – signal detect
13	Sec. request to send
14	Sec. received data
15	Transmitter signal timing (DEC source)
16	Sec. transmitted data
17	Receiver signal timing (DEC source)
18	Local loopback
19	Sec. clear to send
20	DTE ready
21	Remote loopback
22	Ring indicator
23	Data signal rate selector
24	Transmitter signal timing (DTE source)
25	Test mode

Tabela 4. Pinagem para um sistema DCE, conector DB-9(fêmea)

Pino	Descrição
1	Received line signal detect
2	Transmitted data
3	Received data
4	DTE ready
5	Signal ground
6	DCE ready
7	Clear to send
8	Request to send
9	Ring indicator

Os nomes dos sinais que implicam em direção, como por exemplo, *Transmit Data* e *Received Data*, devem ser nomeados do ponto de vista do dispositivo que se está fazendo a conexão. O *transmit data* de um dispositivo (DTE) com o *received data* do outro dispositivo (DCE). A Figura 1 ilustra a convenção utilizada para os sinais mais comuns.

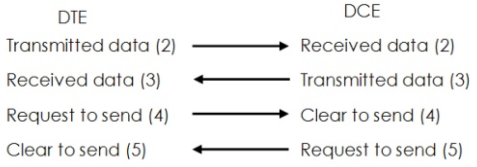

Fig. 1. Convenção de conexão para os sinais comuns do padrão RS-232.

Descrição das pinagens

A descrição dos sinais segundo o padrão RS-232 é mostrada na tabela 5, dividido por categorias.

Tabela 5. Descrição da pinagem para o padrão RS-232

Pino	Nome	Descrição
	Sinais de terra	
1	Shield	Sinal de terra de proteção (malha de aterramento do cabo e carcaça do conector)
7	Ground (GND)	Sinal de terra utilizado como referência
	Canal de comunicação primário	
2	Transmitted data (TxD)	Sinal transmitido do DTE para o DCE ou do DCE para o DTE
3	Received data (RxD)	Sinal recebido do DTE para o DCE ou do DCE para o DTE
4	Request to send (RTS)	Sinal é habilitado (nível lógico 0) para preparar o DCE para aceitar os dados transmitidos pelo DTE.
5	Clear to send (CTS)	Sinal é habilitado (nível lógico 0) pelo DCE para informar ao DTE que a transmissão pode começar.
	Canal de comunicação secundário	
14	Secondary transmitted data (STxD)	Equivalente ao sinal TxD
16	Secondary received data (SRxD)	Equivalente ao sinal RxD
19	Secondary request to send (SRTS)	Equivalente ao sinal RTS
13	Secondary clear to send (SCTS)	Equivalente ao sinal CTS
	Sinais de controle de status do modem	
6	DCE ready (DSR)	Também chamado de Data Set Ready. Quando chamado de um modem, este sinal é habilitado em nível lógico 0, quando as seguintes condições são satisfeitas: 1 – modem conectado a uma linha telefônica ativa e fora do gancho; 2 – modem estiver no modo dados; 3 – modem tiver completado a discagem e está gerando um tom de resposta.
20	DTE Ready (DTR)	Também chamado de Data Terminal Ready. Este sinal é habilitado pelo DTE quando for necessário abrir o canal de comunicação.
8	Received line signal detector (CD)	Também chamado de Data Carrier Detect (DCD). Mais usado quando o DCE é um modem
12	Secondary Received Line Signal Detector (SCD)	Equivalente ao CD
22	Ring indicator (RI)	Identifica o sinal de chamada da linha telefônica.
23	Data Signal Rate Selector	Seleciona um de dois *baud rates* pré-configurados.
	Sinais de transmissão e recepção de tempos	
15	Transmitter Signal Element Timing (TC)	Transmitter Colck (TxC). O modem gera este sinal de clock para controlar exatamente a taxa na qual os dados estão sendo enviado pelo pino TxD do DTE para o DCE.
17	Receiver Signal Element timing (RC)	Receiver Clock (RxC). Similar ao TC.
24	Transmitter Signal Element Timing (ETC)	Usado apenas quando os sinais TC e RC não estão sendo utilizados
	Sinais de teste do canal de comunicação	
18	Local Loopback (LL)	Gerado pelo DTE. Os sinais são ecoados pelo modem.
21	Remote loopback (RL)	Gerado pelo DTE. O sinal é ecoado pelo modem remoto.
25	Test Mode (TM)	Indica que o modem esta em condição de teste local ou remoto.

Padrão do sinal da RS-232

O padrão de sinal utilizada para a comunicação RS-232 compreende três seqüências, se o sinal de tensão esta entre -3 a -25Vdc, em relação ao pino terra (7), é considerado nível lógico 1, entre +3 e +25Vdc é considerado nível lógico 0. A faixa entre -3 a +3Vdc é considerado uma região de transição.

Na Figura 2 é ilustra os níveis de sinais para um padrão RS-232.

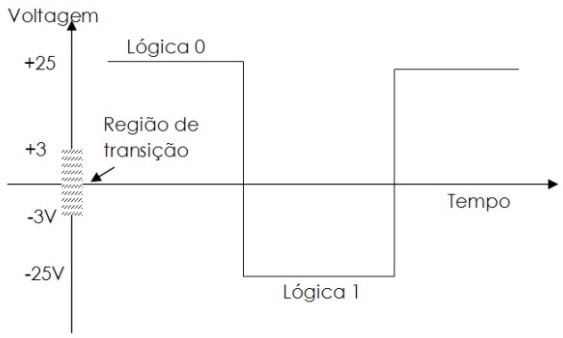

Fig. 2. Padrão do nível de sinal RS-232.

Temporização de sinais

Os valores de taxa de transferência de dados estabelecidos pela norma EIA232 mais comumente usados são: 300; 1200; 4800; 9600 e 19200 bps.

Conversores TTL para Rs232

A maioria dos sistemas digitais trabalham com tensões de 5Vdc, representando o nível lógico 1, e tensão de 0Vdc, para representar o nível lógico 0. Assim, para se conectar um sistema digital a uma interface RS232 é necessário fazer uma transformação destes níveis lógicos para o correspondente padrão RS232.

Muitos fabricantes de circuitos integrados têm colocado no mercado conversores de níveis TTL para RS232 e um dos mais comuns utilizados é o circuito integrado MAX232 de fabricação da MAXIM (www.maxim-ic.com) como da TEXAS Instruments (ww.ti.com).

Este circuito gera tensões de +10 Vdc e 10 Vdc a partir de uma alimentação de 5Vdc, uma configuração típica deste circuito é ilustrada na Figura 3.

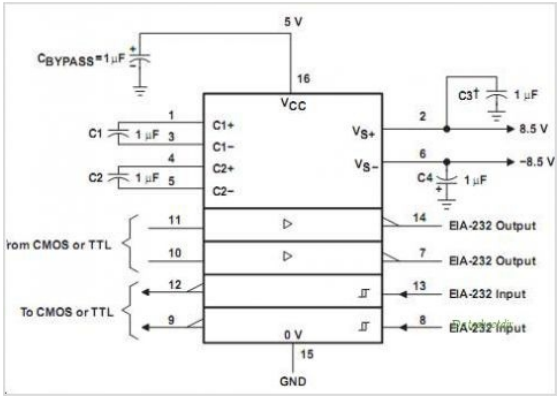

Fig. 3. Esquema de conexões do circuito integrado RS232 (Texas Instruments).

[1] www.sun.com ou br.sun.com

Configuração geral de um cabo serial a três fios.

Na tabela 6 é ilustrado o esquema de ligação para um cabo de comunicação serial utilizando três fios. Este tipo de cabo é muito utilizado para fazer a comunicação serial via dois computadores PC.

Tabela 6. Conexão cabo serial a três fios.

Conector DB25	Conector DB9	Sinal		Sinal	Conector DB9	Conector DB25
2	3	TD	→	RD	2	3
3	2	RD	←	TD	3	2
7	5	SG	↔	SG	5	7
20	4	DTR	→	DTR	4	20
6	6	DSR	←	DSR	6	6
8	1	CD	←	CD	1	8
4	7	RTS	→	RTS	7	4
5	8	CTS	←	CTS	8	5

Programa JAVA para comunicação serial.

Para que um programa em linguagem Java possa rodar em um computador, é necessário fazer a instalação dos aplicativos JDK e JRE (java developer kit e java runtime enveiroment).

Ambas podem ser baixadas diretamente do sítio da SUN Microsystem nas versões mais atualizadas. A versão utilizada neste documento é a JDK6.

Ao ser instalado, o programador deve colocar no PATH o caminho onde foram instaladas as versões.

Uma vez instalados os aplicativos, os programas desenvolvidos em linguagem Java são compilados por meio do comando de linha "javac" e executado pelo comando de linha "java", por exemplo:

Compilação: javac nomedoprograma.java
Execução: java nomedoprograma

A edição do programa pode ser feita com qualquer editor de texto, desde que seja salvo com a extensão "*.java" .

Estes programas são executados através de comandos de linha em uma janela do Windows denominada "Prompt de Comando" (tela DOS do Windows).

O método de se editar o programa em um editor de texto, exige do programador um conhecimento muito bom na linguagem Java.

O problema disto é que é mais trabalhoso, ou seja, montar o programa; editar; compilar; verificar se não há erros e por fim executar. Contornando isto, a SUN disponibiliza um aplicativo que facilita bastante a idealização de um programa em JAVA, que é um programa com uma interface gráfica chamada NETBEANS. Este aplicativo também pode ser baixado gratuitamente do sítio da SUN Microsystem[1].

O NetBeans facilita muito a vida do programador, apresentando em forma gráfica e com cores os diferentes comandos de linguagem Java, e indicando possíveis erros no decorrer da edição do programa.

Ao se optar pela instalação do NetBeans, não é necessário a instalação dos aplicativos JDK e JRE. Pois quando se está instalando o NetBeans, o programa de instalação já coloca as versões mais atualizadas destes aplicativos, uma vez que são requisitos obrigatórios para a execução do NetBeans.

Na Figura 4 é representado o ambiente de desenvolvimento do NetBeans.

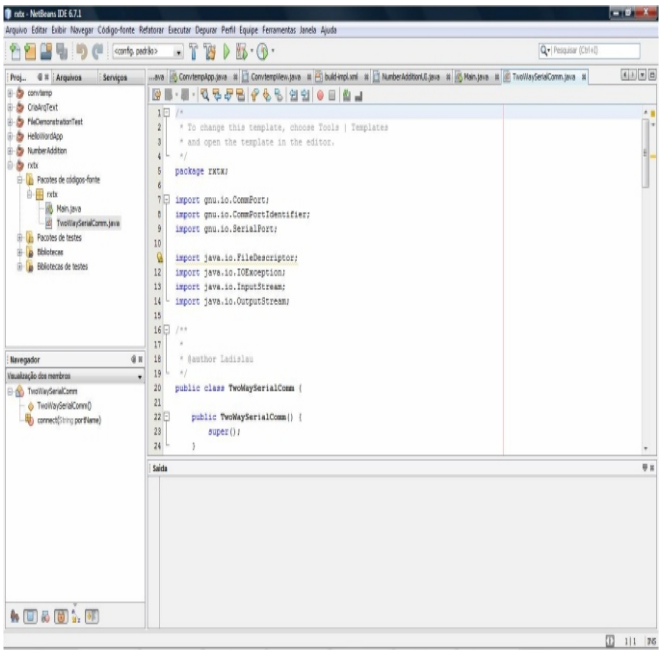

Fig. 4. Ambiente de desenvolvimento do aplicativo NetBeans.

O NetBeans vem com um grande número de bibliotecas que auxiliam em muito os programadores, porém não vem com as bibliotecas para comunicação serial.

Esta biblioteca tem que ser baixada e instalada conforme os procedimentos anteriores. São encontradas no sítio RxTx[2] de onde pode ser baixado o arquivo rxtx-2.1-7-bins-r2, que está compactado em um arquivo "zip" e que é a atual versão no momento deste documento.

Ao ser descompactado este mostra os seguintes arquivos, conforme ilustrado na Figura 5.

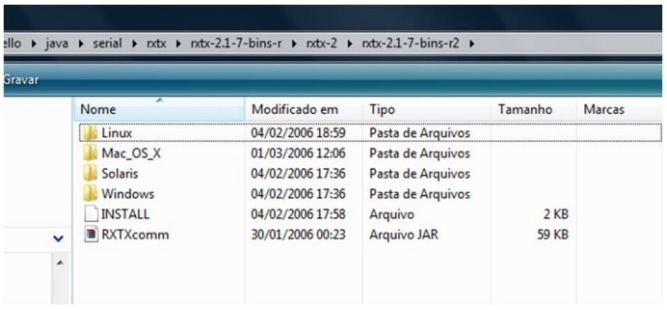

Fig. 5. Arquivos descompactados do arquivo rxtx-2.1-7-bins-r2.

Nota-se na Figura 5 que o programa foi desenvolvido para vários sistemas operacionais. No nosso caso este foi usado para o sistema operacional Windows, mais precisamente no Windows Vista Basic Home.

Abrindo a pasta Windows temos os seguintes arquivos, conforme ilustrado na Figura 6.

Copie a DLL rxtxSerial nos diretórios de instalação do programa JAVA, que deve ter sido criado e instalado no momento da instalação do NetBeans.

No diretório JDK1.6.0_14, copie a DLL rxtxSerial na pasta "jre\bin" e o mesmo para o diretório jre1.6.0_05 na pasta "bin". As versões 1.6 indicadas aqui são as mais atuais durante a edição deste documento.

[2]http://rxtx.qbang.org

O arquivo RXTXcomm deve ser copiado também no diretório jre1.6.0_05 na pasta "\lib\ext" .

Fig. 6. Arquivos da pasta Windows.

Listagem do programa JAVA

Ao iniciar o aplicativo NETBEANS e se não houver um projeto em andamento, a tela inicial é mostrada na Figura 7.

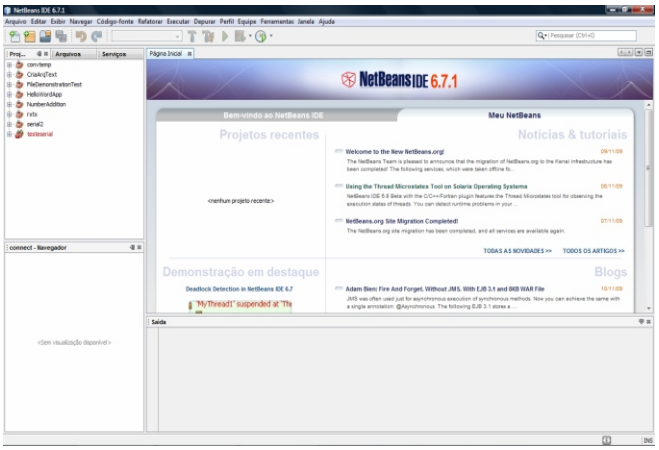

Fig. 7. Janela inicial do aplicativo NETBEANS.

Para fazer um novo programa, abre-se o menu "arquivo" e seleciona "novo projeto" (Fig. 8), ao se selecionar "novo projeto" uma nova janela é exibida, onde escolhemos "Categoria: Java" e Projetos: Aplicativo Java" (Fig. 9).

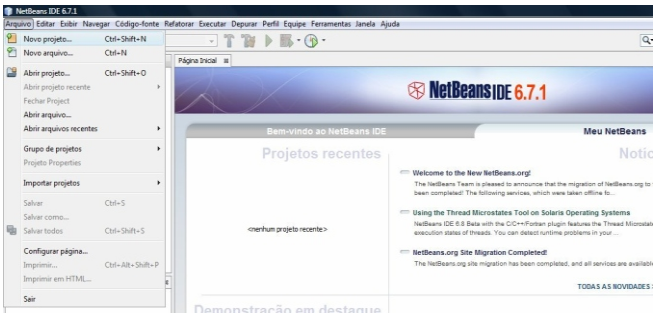

Fig. 8: Menu arquivo para selecionar novo projeto.

Com o botão esquerdo do mouse, seleciona-se o botão "próximo", que abrirá outra tela (Fig. 10). Em "nome do projeto", escolhemos um nome e em "localização do projeto" o lugar onde será armazenado e finalmente o botão "finalizar".

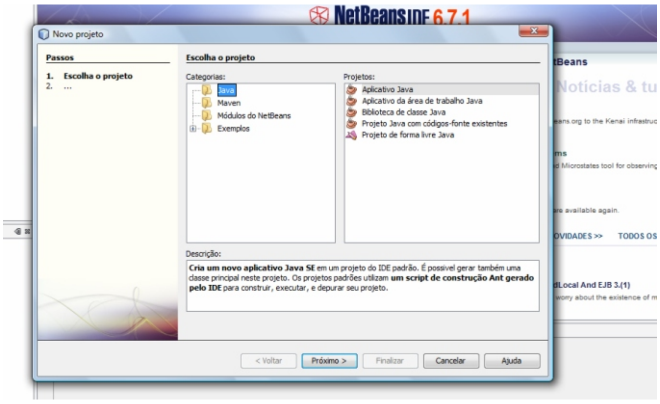

Fig. 9. Tela de seleção de novo projeto.

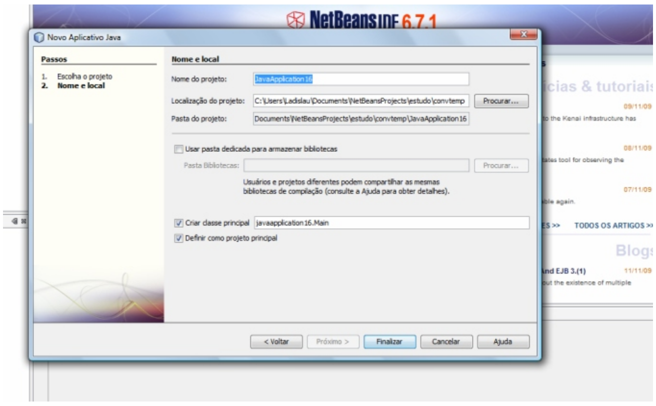

Fig. 10. Tela para nome e local do novo projeto.

O aplicativo NETBEANS agora apresenta uma nova tela (Fig. 11). Nesta tela temos a esquerda o nome e alguns dados do projeto e a direita, em destaque a página onde será escrito o programa.

A listagem do programa deve ser iniciada após a linha " *// TODO code application logic here*", dentro da classe "*Main*".

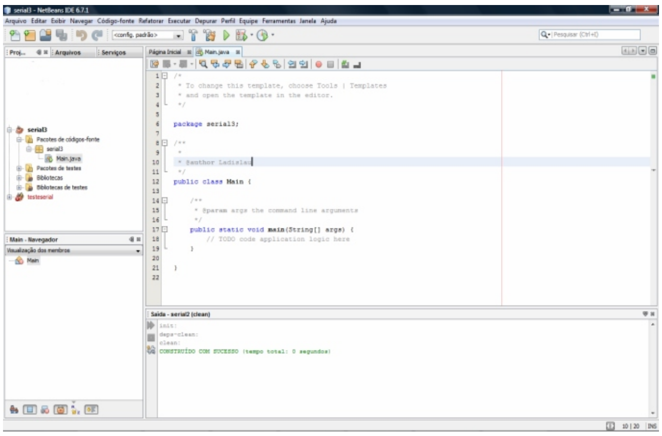

Fig. 11. Tela de inicio de escrita do programa.

Para facilitar um pouco a programação, dividimos o programa em duas classes, a principal como indicado na Figura 11, que executará outra classe onde será escrito o programa em Java para a comunicação serial.

O programa foi deixado da mesma maneira como se encontra o exemplo no sítio da RxTx[3].

Para criar a outra classe, localiza-se o ponteiro do mouse sobre o nome do projeto localizado no item "Pacotes de códigos-fonte" e aciona-se o botão direito do mouse, seleciona-se "novo" e depois "classe java" (Fig. 12).

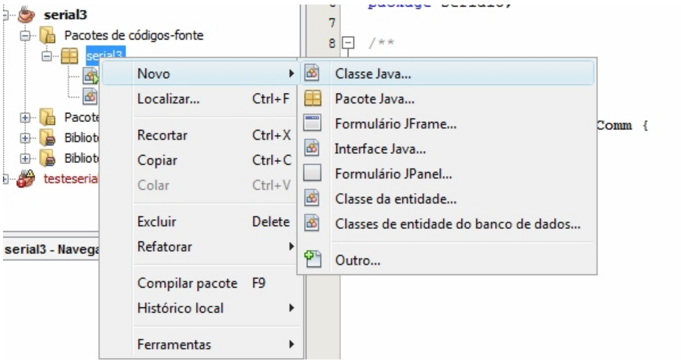

Fig. 12. Criando uma nova classe java.

Em seguida na janela "Nova Classe Java" escolher um nome e selecionar o botão "Finalizar" (Fig. 13).

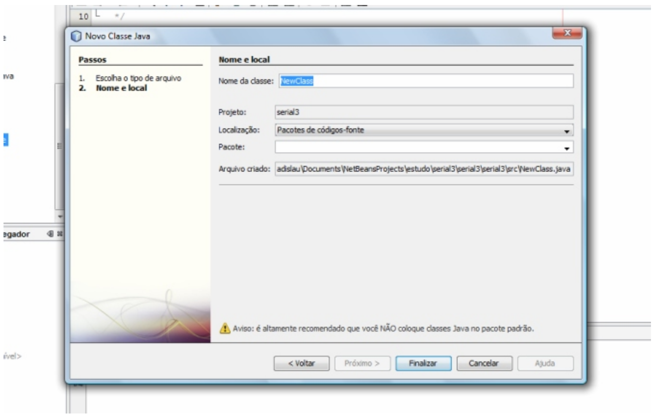

Fig. 13. Janela "Nova Classe Java".

Ao finalizar o aplicativo, o NetBeans retorna à sua pagina inicial conforme ilustrada na Figura 11.

Nesta página, logo após a linha "//TODO....", escreveremos o seguinte código de linguagem java:

= =

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package serial3;
/**
 *
 * @author ***********
 */
public class Main {
   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
       // TODO code application logic here
       try {
           (new TwoWaySerialComm()).connect("COM4");
             }catch (Exception e) {
            e.printStackTrace();
          }
      }
}
```

= =

[3]http://rxtx.qbang.org/wiki/index.php/Two_way_communcation_with_the_serial_port

Obs. colocar aqui a porta serial que será usada "COM1; COM2; COM3 ou COM4".

Selecionando a nova classe, com um duplo toque no botão esquerdo do mouse é selecionada a página para digitar o seguinte código em linguagem Java:

```
================================================================
=
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package serial3;

import gnu.io.CommPort;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;

import java.io.FileDescriptor;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
/**
 *
 * @author ***********
 */
public class TwoWaySerialComm {
    public TwoWaySerialComm() {
        super();
    }
    void connect(String portName) throws Exception {
                        CommPortIdentifier portIdentifier =
CommPortIdentifier.getPortIdentifier(portName);
        if (portIdentifier.isCurrentlyOwned()) {
            System.out.println("Error: `port is currently in use");
        }else {
                                    CommPort commPort =
portIdentifier.open(this.getClass().getName(), 2000);
            if (commPort instanceof SerialPort) {
                SerialPort serialPort = (SerialPort) commPort;

serialPort.setSerialPortParams(9600,SerialPort.DATABITS_8,SerialPort.STO
PBITS_1,SerialPort.PARITY_NONE);
                InputStream in = serialPort.getInputStream();
                OutputStream out = serialPort.getOutputStream();
                (new Thread(new SerialReader(in))).start();
                (new Thread(new SerialWriter(out))).start();
            }else {
                    System.out.println("Error: only serial ports are handled by this
example.");
            }
        }
    }
    public static class SerialReader implements Runnable {
        InputStream in;
        public SerialReader(InputStream in) {
            this.in = in;
        }
        public void run() {
            byte[] buffer = new byte[1024];
            int len = -1;
            try {
                while ((len = this.in.read(buffer)) > -1) {
                        System.out.print(new String(buffer, 0, len));
                }
            }catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    public static class SerialWriter implements Runnable {
        OutputStream out;
        public SerialWriter(OutputStream out) {
            this.out = out;
        }
        public void run() {
            try {
                int c = 0;
                    while ((c = System.in.read()) > -1) {
                    this.out.write(c);
                }
               }catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
================================================================
=
```

Uma vez escrito todo o código fonte para as classes Main e TwoWaySerialComm, executamos o comando de construir o programa através da tecla de atalho F11, ou no menu "Executar construir projeto Main".

Se tudo estiver normal e escrito correto o compilador apresenta uma mensagem de sucesso de construção do projeto.

Para executar o programa podemos usar as teclas de atalho F6 ou no menu "Executar executar projeto" . Na tela do aplicativo NetBeans, abaixo a direita surge a seguinte mensagem:

```
run:
Stable Library
================================================
Native lib Version = RXTX-2.1-7
Java lib Version   = RXTX-2.1-7
```

Na parte inferior a direita uma faixa indicando que o programa esta em execução, conforme ilustrado na Figura 14.

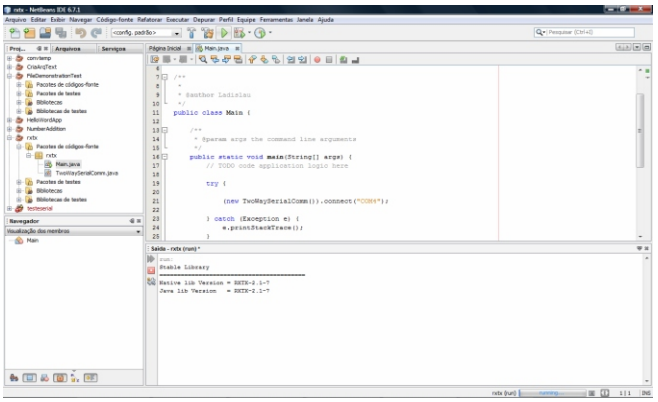

Fig. 14. Tela de execução do programa serial.

Logo abaixo da mensagem escrever o que se deseja enviar via serial e pressionar a tecla "enter" do microcomputador.

Para parar a execução do programa, selecionar a cruz logo a esquerda da faixa na parte inferior a direta da tela.

Comentário final

O programa aqui apresentado é de uso livre, desenvolvido pela RxTx com suas bibliotecas, e o seu uso restringe-se as normas de programas livres.

Literatura consultada

MAXIM INTEGRATED PRODUCTS. **Maxim**: +5v-powered, multichannel RS-232 drivers/receivers. San Gabriel Drive, c2006. Disponível em: <http://datasheets.maxim-ic.com/en/ds/MAX220-MAX249.pdf>. Acesso em: 24 nov. 2009. p. 19-4323; Rev 15; 1/06.

NETBEANS.ORG. **Java**. [2009]. Disponível em: <http://netbeans.org/downloads/index.html>. Acesso em: 24 nov. 2009.

RXTX. Disponível em: http://rxtx.qbang.org. Acesso em: 24 nov. 2009.

SERIAL PROGRAMMING. Disponível em: http://en.wikibooks.org/wiki/Serial_Programming/Serial_java. Acesso em: 24 nov. 2009.

STRANGIO, C. E. **The RS232 standard**: a tutorial with signal names and definitions. c 2006. Disponívem em: <http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html 24 nov. 2009>. Acesso em: 24 nov. 2009.

SUN MICROSYSTEMS INCORPORATED. **Java**. c2009. Disponível em: <http://java.com/pt_BR/download/windows_xpi.jsp?locale=pt_BR&host=java.com>. Acesso em: 24 nov. 2009.

TEXAS INSTRUMENTS INCORPORATED. **Max 232**. Dallas, c2009. Disponível em: <http://focus.ti.com/lit/ds/symlink/max232.pdf>. Acesso em: 24 nov. 2009.

Comunicado Técnico, 109

Exemplares desta edição podem ser adquiridos na:
Embrapa Instrumentação Agropecuária
Rua XV de Novembro, 1542 - Caixa Postal 741
CEP 13560-970 - São Carlos-SP
Fone: 16 2107 2800 - **Fax:** 16 2107 2902
e-mail: sac@cnpdia.embrapa.br
http://www.cnpdia.embrapa.br
1a. edição
1a. impressão 2009: tiragem 300

Ministério da Agricultura, Pecuária e Abastecimento
BRASIL UM PAÍS DE TODOS GOVERNO FEDERAL