Comunicação Serial Java + Arduino

por Klauder Dias em sexta-feira, 14 de fevereiro de 2014.

Introdução

Neste artigo demonstrarei como ligar e desligar um led, através de comandos enviados pela porta serial (USB) utilizando Java e uma placa Arduino.

Para ler outro artigo que aborda a comunicação serial clique aqui para ler o artigo de Fábio Souza, e caso queira saber mais sobre programação no arduino clique aqui.

Mas por que utilizar *Universal Serial Bus* (USB)?

A comunicação USB é uma tecnologia que tornou mais simples e rápida a conexão de diversos tipos de aparelhos (Pendrives,MP3-players, impressoras, celulares, HDs externos, etc). Além de facilitar a conexão de diversos dispositivos, o padrão USB oferece:

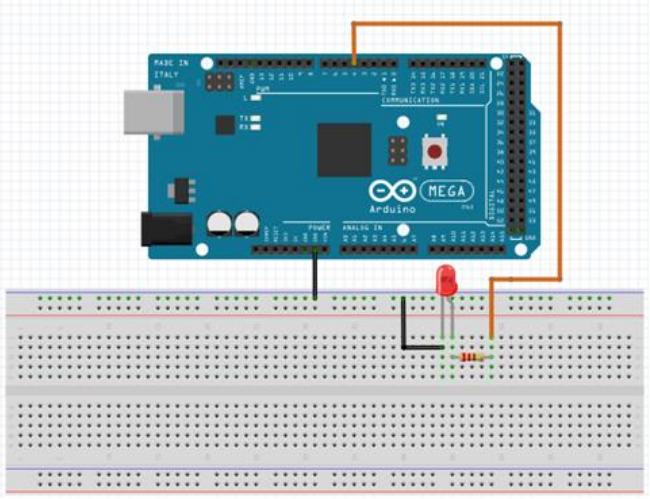
- **Plug-and-play**: a maioria dos dispositivos são reconhecidos automaticamente;
- **Alimentação**: os dispositivos não precisam ser ligados a uma fonte de energia externa, já que a própria USB fornece energia. (Lembrado a USB fornece no máximo 5 Volts e 500 mA de corrente);
- **Múltipla conexão:** vários dispositivos podem ser conectados através de uma única porta, mas para isso deve-se utilizar um *Hub;*
- **Compatibilidade:** são compatíveis com a maioria dos sistemas operacionais disponíveis no mercado.

A comunicação entre os sistemas embarcados e os computadores muitas vezes são realizadas via porta paralela (DB25) e porta serial (DB-9). A utilização desses meios de comunicação pode facilitar uma alteração futura mas, por outro lado, dependendo do dispositivo, a implementação do software no computador pode ser complexa e trabalhosa. Além do mais, podemos danificar permanentemente a placa mãe do computador quando estamos utilizando a porta paralela e ocorre uma descarga elétrica ou invertermos a polaridade de um componente.

Esse artigo foi elaborado utilizado o Arduino Mega 2560, *NetBeans* 7.4, Windows 7 *Professional* x64 e Ubuntu 12.04 LTS (x64),

estou considerando que as IDE's Arduino 1.0.5 ou superior e *NetBeans* 7.4 ou superior já estejam instaladas e funcionando corretamente.

Obs.: Pode-se utilizar também o Arduino Uno, ADK, Pro.



Passo 01: Download dos arquivos

Acesse o site http://jlog.org/rxtx.html e faça download dos arquivos rxtxSerial.dll, librxtxSerial.so e RXTXcomm.jar.

Obs.: Lembrando de que devemos fazer download dos arquivos de acordo a arquitetura do sistema operacional 32-bits (x86) ou 64-bits (x64).

Passo 02: Instalação dos arquivos

Windows

Copie o arquivo ***rxtxSerial.dll*** para:

- C:\Program Files\Java\jdkx.x.x\bin, onde x.x.x é a versão do JDK, por exemplo C:\Program Files\Java\jdk1.6.40\bin;
- C:\Program Files\Java\jrex\bin, onde x é a versão do JRE, por exemplo C:\Program Files\Java\jre7\bin;
- C:\Windows\System32;
- C:\Windows\SysWOW64 (caso sistema operacional 64-bits (x64)).

Copie o arquivo ***RXTXcomm.jar*** para:

- C:\Program Files\Java\jre\jrex\lib\ext, onde x é a versão do JRE, por exemplo C:\Program Files\Java\jre7\lib\ext.

Linux

Copie o arquivo ***librxtxSerial.so*** para:

- /usr/lib/, exemplo: cp /home/Usuario/librxtxSerial.so /usr/lib/.

Copie o arquivo RXTXcomm.jar para:

- /usr/share/java/, exemplo.: cp /home/Usuario/RXTXcomm.jar /usr/share/java/.

Passo 03: Programa Arduino

Conecte o cabo USB no Arduino e abra a IDE do Arduino, defina a porta COM (*Tools > Serial Port > COM3 ou /dev/ttyUSB0*), digite o código abaixo e depois clique no upload. Pronto, o programa já foi carregado no arduino.

Obs.: Geralmente utiliza-se a porta COM3 (Windows) e /dev/ttyUSB0 (Linux), caso apareça outra porta não tem problema, desde que a porta correta seja informada no programa em Java.

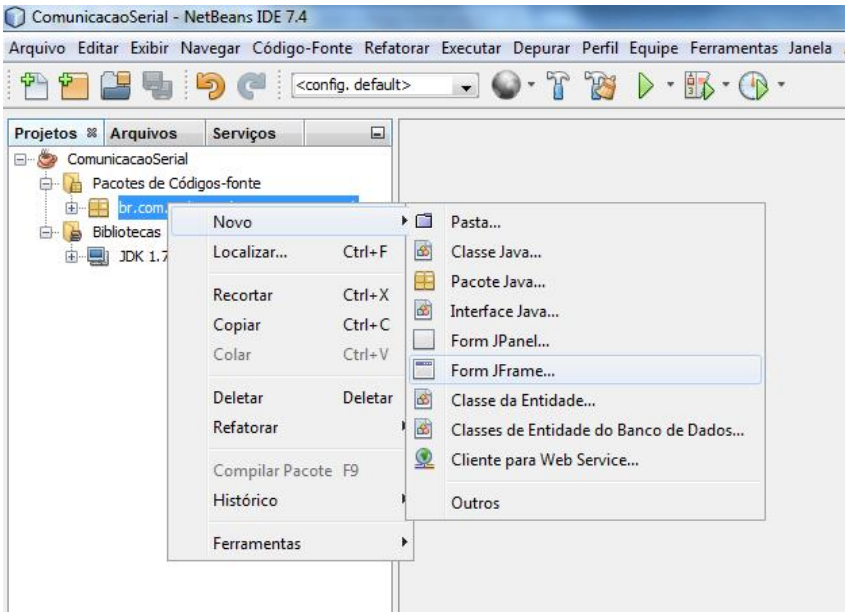
```
int ledPin =  13; //atribui o pino 13 a variável ledPin
int dado; //variável que receberá os dados da porta serial

void setup(){
  Serial.begin(9600);//frequência da porta serial
   pinMode(ledPin,OUTPUT); //define o pino o ledPin como saída
}

void loop(){
  if(Serial.available() > 0){ //verifica se existe comunicação com a porta serial
      dado = Serial.read();//lê os dados da porta serial
      switch(dado){
        case 1:
           digitalWrite(ledPin,HIGH); //liga o pino ledPin
        break;
        case 2:
           digitalWrite(ledPin,LOW); //desliga o pino ledPin
         break;
    }
  }
}
```

Passo 04: Programa Java

Abra o programa *NetBeans* 7.4 criei um projeto Aplicação Java (Arquivo > Novo Projeto > Java > Aplicação Java) e vamos chamá-lo de ***ComunicacaoSerial***.

Renomeie o pacote ***"comunicacaoserial"*** para ***"br.com.embarcados.comunicaoserial"*** e depois apague a classe ComunicacaoSerial.java que foi gerada.

Feito isso, adicione um JFrame no pacote ***"br.com.embarcados.comunicaoserial"*** (botão direito em cima do pacote > Novo > Form Frame), e chame-o de JFInterface.



Na Paleta de controle de ***Swing*** *(lado direito da tela)*, clique em cima do botão e arraste para dentro do JFInterface. Após esse passo altere o nome da variável *jButtom1* para *jBLenOn* e altere o texto do botão para Ligar. Basta clicar com o botão direito em cima do botão e escolher as opções Editar Texto e Alterar nome da variável.

Copie mais 02 botões para o JFInterface e repita o mesmo processo renomeando-os para ID: jBLenOff, jBClose e Text: Desligar e Sair.

Acrescente duas classes no projeto “Arduino.java e ControlePorta.java” (botão direito em cima do pacote > Novo > Classe Java.

No arquivo Arduino.java realizaremos o envio das informações para o Arduino e no arquivo ControlePorta.java faremos toda a programação para enviar os dados através da porta serial.

Adicione o código abaixo no arquivo ControlePorta.java.

```
1  package br.com.embarcados.comunicacaoserial;
2
3  import gnu.io.CommPortIdentifier;
4  import gnu.io.NoSuchPortException;
5  import gnu.io.SerialPort;
6  import java.io.IOException;
7  import java.io.OutputStream;
8  import javax.swing.JOptionPane;
9
10 public class ControlePorta {
11   private OutputStream serialOut;
12   private int taxa;
13   private String portaCOM;
```

```
14
15   /**
16    * Construtor da classe ControlePorta
17    * @param portaCOM - Porta COM que será utilizada para enviar os dados para o arduino
18    * @param taxa - Taxa de transferência da porta serial geralmente é 9600
19    */
20   public ControlePorta(String portaCOM, int taxa) {
21     this.portaCOM = portaCOM;
22     this.taxa = taxa;
23     this.initialize();
24   }
25
26   /**
27    * Médoto que verifica se a comunicação com a porta serial está ok
28    */
29   private void initialize() {
30     try {
31       //Define uma variável portId do tipo CommPortIdentifier para realizar a comunicação serial
32       CommPortIdentifier portId = null;
33       try {
34         //Tenta verificar se a porta COM informada existe
35         portId = CommPortIdentifier.getPortIdentifier(this.portaCOM);
36       }catch (NoSuchPortException npe) {
37         //Caso a porta COM não exista será exibido um erro
38         JOptionPane.showMessageDialog(null, "Porta COM não encontrada.",
39                   "Porta COM", JOptionPane.PLAIN_MESSAGE);
40       }
41       //Abre a porta COM
42       SerialPort port = (SerialPort) portId.open("Comunicação serial", this.taxa);
43       serialOut = port.getOutputStream();
44       port.setSerialPortParams(this.taxa, //taxa de transferência da porta serial
45                                SerialPort.DATABITS_8, //taxa de 10 bits 8 (envio)
46                                SerialPort.STOPBITS_1, //taxa de 10 bits 1 (recebimento)
47                                SerialPort.PARITY_NONE); //receber e enviar dados
48     }catch (Exception e) {
49       e.printStackTrace();
50     }
51 }
52
53   /**
54    * Método que fecha a comunicação com a porta serial
55    */
56   public void close() {
57     try {
58         serialOut.close();
59     }catch (IOException e) {
60       JOptionPane.showMessageDialog(null, "Não foi possível fechar porta COM.",
61                 "Fechar porta COM", JOptionPane.PLAIN_MESSAGE);
62     }
63   }
64
65   /**
66    * @param opcao - Valor a ser enviado pela porta serial
67    */
68   public void enviaDados(int opcao){
69     try {
70       serialOut.write(opcao);//escreve o valor na porta serial para ser enviado
71     } catch (IOException ex) {
72         JOptionPane.showMessageDialog(null, "Não foi possível enviar o dado. ",
73                 "Enviar dados", JOptionPane.PLAIN_MESSAGE);
74     }
75   }
76 }
```

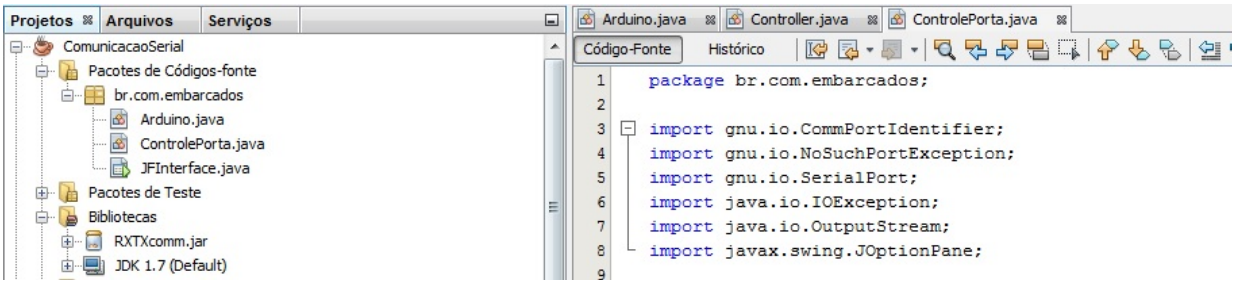
Após adicionar o código acima acontecerá vários erros conforme a mensagem abaixo.

```
1  package br.com.embarcados;
2
3  import gnu.io.CommPortIdentifier;
4  import gnu.io.NoSuchPortException;
5  import gnu.io.SerialPort;
6  import java.io.IOException;
7  import java.io.OutputStream;
8  import javax.swing.JOptionPane;
9
```

Esses erros aconteceram porque não adicionamos o arquivo RXTXcomm.jar no nosso projeto. Para adicioná-lo clique com o botão direito em cima da pasta Biblioteca e escolha Adicionar JAR/Pasta... e informe o diretório onde o arquivo RXTXcomm.jar

(C:\Program Files\Java\jre7\lib\ext\RXTXcomm.jar, caso esteja utilizando Windows) ou (/usr/share/java/ RXTXcomm.jar, caso esteja utilizando Linux).

Limpe e construa o projeto (botão direito em cima do projeto ComunicacaoSerial > Limpar e Construir) e os problemas serão corrigidos conforme pode-se ver na imagem abaixo.



Agora adicione o código abaixo no arquivo Arduino.java.

```
1  package br.com.embarcados.comunicacaoserial;
2
3  import javax.swing.JButton;
4
5  /**
6   * @author klauder
7   */
8  public class Arduino {
9    private ControlePorta arduino;
10
11   /**
12    * Construtor da classe Arduino
13    */
14   public Arduino(){
15       arduino = new ControlePorta("COM3",9600);//Windows - porta e taxa de transmissão
16       //arduino = new ControlePorta("/dev/ttyUSB0",9600);//Linux - porta e taxa de transmissão
17   }
18
19   /**
20    * Envia o comando para a porta serial
21    * @param button - Botão que é clicado na interface Java
22    */
23   public void comunicacaoArduino(JButton button) {
24     if("Ligar".equals(button.getActionCommand())){
25       arduino.enviaDados(1);
26       System.out.println(button.getText());//Imprime o nome do botão pressionado
27     }
28     else if("Desligar".equals(button.getActionCommand())){
29       arduino.enviaDados(2);
30       System.out.println(button.getText());
31     }
32     else{
33       arduino.close();
34       System.out.println(button.getText());//Imprime o nome do botão pressionado
35     }
36   }
37 }
```

Feito isso, falta implementar a ação a ser executada quando um determinado botão for pressionado.

Abra o arquivo JFInterface.java, clique com o botão direito em cima do botão “Ligar” e acione o método MousedClicked. Será criado automaticamente um método dentro da classe JFInterface.java conforme a imagem abaixo.

```
private void jBLedOnMouseClicked(java.awt.event.MouseEvent evt) {
  // TODO add your handling code here:
}
```

Declare uma variável do tipo Arduino antes do construtor da classe JFInterface, conforme a figura abaixo.

```
7   package br.com.embarcados.comunicacaoserial;
8
9   import java.awt.event.MouseAdapter;
10  import java.awt.event.MouseEvent;
11
12  /**
13   *
14   * @author klauder
15   */
16  public class JFInterface extends javax.swing.JFrame {
17
18      Arduino conn = new Arduino();
19
20    /**
21       * Creates new form JFInterface
22       */
23      public JFInterface() {
24          initComponents();
25      }
```

Dentro do método jBLedOnMouseClicked criado anteriormente adicione o código conn.comunicacaoArduino(jBLedOn).

```
private void jBLedOnMouseClicked(java.awt.event.MouseEvent evt) {
  // TODO add your handling code here:
  conn.comunicacaoArduino(jBLedOn); //Nome do botão pressionado
}
```

Esse comando irá acionar o método comunicacaoArduino(), passando o botão clicado, que por sua vez irá enviar o dado pela porta serial.

Adicione o mesmo evento para o botão “Desligar” e coloque o código abaixo:

```
private void jBLedOffMouseClicked(java.awt.event.MouseEvent evt) {
  // TODO add your handling code here:
  conn.comunicacaoArduino(jBLedOff);
}
```

Adicione o mesmo evento para o botão “Sair” e coloque o código abaixo:

```
private void jBCloseMouseClicked(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:
  conn.comunicacaoArduino(jBClose);
  System.exit(0);//fecha a aplicação
}
```

Passo 05: Executando o programa

Agora basta apenas conectar o seu Arduino à porta USB do seu computador e clicar em Executar o projeto Java. Ao aparecer a Tela, basta apenas clicar no botão “Ligar” para ligar, “Desligar” para desligar o Led e “Sair” para encerrar aplicação.

Os códigos fontes utilizados na elaboração desse artigo estão disponíveis em: **https://github.com/klauder/ComunicacaoSerial**

Espero que tenham gostado e até breve.

Referências

http://jlog.org/rxtx.html

http://arduino.cc/en/Main/ArduinoBoardMega2560

http://arduino.cc/en/reference/serial

Klauder Dias

Pós-graduando em Gerenciamento de Projetos. Graduado em Engenharia de Computação. Atualmente trabalha com Analista de desenvolvimento. Experiência em integração de sistemas (hardware/software) utilizando C++ e Java.

Posts Similares

Placa Arduino da Robocore – BlackBoard

A placa BlackBoard, é uma placa Arduino compatível fabricada pela empresa RoboCore aqui no Brasil. É bem similar às placas Arduino Uno R3 e Duemilanove, com...
veja+

Codificação segura em C para sistemas embarcados

O evento TDC2013 São Paulo (The Developer's Conference) contou com diversas trilhas sobre assuntos diversos, tal como Agile, Cloud, games, testes, etc. Além dessas, foram realizadas...
veja+

Arduino – Controle de uma lâmpada com LDR

No artigo anterior apresentamos como ligar uma lâmpada AC utilizando um relê. Dando continuidade a esse assunto, agora vamos explicar como acionar automaticamente uma lâmpada conforme...
veja+

Comentários

Fernando Moreira • segundos atrás

Primeiro parabéns ao colega pelo post, sensacional. Vai uma dica para a galera que usa Linux, a porta ACM0 de alguma forma fica ocupada nesse processo, ai não conseguimos acessá-la, vai dar erro na identificação da porta que vai voltar como null, depois de muito debug e muita pesquisa consegui resolve ro problema.

Basta criar um link simbólico para a porta e referenciá-la através do link:

ln -sf /dev/ttyACM0 /dev/ttyUSB1

Achei essa solução nesta página: http://blog.avisi.nl/2013/01/2...

Tiago Monteiro • 6 dias atrás

Sr. Klauder. Realizei a implementação dos códigos deste tutorial, a comunicação funcionou perfeitamente. Gostaria de receber também os códigos para realizar a leitura das entradas do Arduino, estou construindo um projeto que fará o acionamento de um motor elétrico, com inversão na rotação e leitura dos valores de corrente.

renan moreira • 14 dias atrás

Minha conexão do Arduino com o java está funcionando mais eu não estou consseguindo pegar a impressão do arduino e receber no java, como faço para receber esses dados do arduino ?

Natália Leite Vieira • 2 meses atrás

Ola...Obrigado pelas informações, acredito que a minha dúvida seja até parecido com o pessoal aqui de baixo. Estou fazendo um projeto onde o Arduino fará uma integração com um Website .... o Arduino terá um sensor de presença, e essa informação será direcionada para o Website que estamos desenvolvendo...você sabe se essa integração é complexa??

klauder Mod → Natália Leite Vieira • 2 meses atrás

Natália, boa tarde.

Se seu website estiver rodando localmente, você pode desenvolver uma aplicação (serviço) que receber o dado do sensor de presença e que de tempos em tempos irá alimentar a base de dados do website.
link de como criar o serviço e instá-lo.
http://msdn.microsoft.com/pt-b...

Caso, o site esteja hospedado na internet você deverá fazer um webservice para poder alimentar a base desse web site com os dados do leitor de presença.

Esse webservice poderá ser acessado de uma aplicação web, ou desktop.

Pesquise sobre WCF (Windown communications fundation) que vai te ajudar a criar o webservice e expor essa funcionalidade.
No endereço https://github.com/klauder/Des... tem um exemplo pronto de WCF.

Oportunidades

- Oportunidade P&D – Macnica – Florianópolis/SC
- Embedded Software Engineer – São Paulo/SP
- Industrial Test Engineer – São Paulo/SP
- Vaga desenvolvimento de hardware Jr. na PHI INNOVATIONS – Campinas / SP
- Programador de embarcados na Writesys – Traffic Systems – Santa Barbara D Oeste/SP

Mais oportunidades
Newsletter

Email

Assinar

Enquetes Realizadas
Escolha de Microcontrolador / Microprocessador
Sistema Operacional Embarcado
Linguagem de programação
Tipo de Processador
O Novo Embarcados

Posts populares
Arduino UNO 6.095 visualizações
BeagleBone Black + Yocto 5.139 visualizações
Simulador de Arduino: Virtual Breadboard 3.920 visualizações
Comunicação Serial Java + Arduino 3.024 visualizações
Links
Sobre
Eventos
Downloads
Login

Contato
Comunidade
Seja Colaborador
Oportunidades
Publicidade
Anuncie Conosco
Fique por dentro
Registre-se
Newsletter

Desenvolvido por beecreative