**PROJETO DE ANÁLISE DE SISTEMAS**

**Etapa - Manual**

**Progeto Agil**

Douglas Vieira

Luis Henrique

Rafael Moog

**Emails:**

[douglascanal1998@gmail.com](mailto:douglascanal1998@gmail.com)

[luisdoperigo24456@gmail.com](mailto:luisdoperigo24456@gmail.com)

[rafaelmoogborges@gmail.com](mailto:rafaelmoogborges@gmail.com)

**Introdução**

Este documento fornece um guia passo a passo sobre como configurar e usar um programa Java para controlar um LED conectado a um Arduino. Usaremos a biblioteca RXTX para facilitar a comunicação serial entre Java e Arduino.

### **Requisitos**

1. **Hardware:**
   * Arduino (qualquer modelo compatível)
   * LED
   * Resistor de 220Ω
   * Protoboard e jumpers
2. **Software:**
   * Arduino IDE
   * Java Development Kit (JDK)
   * Biblioteca RXTX

### **Passo 1: Montagem do Hardware**

1. Conecte o LED e o resistor ao Arduino:
   * Anodo do LED (perna longa) ao pino digital 13 do Arduino.
   * Catodo do LED (perna curta) a um dos terminais do resistor.
   * Outro terminal do resistor ao GND do Arduino.

### **Passo 2: Carregar o Código no Arduino**

1. Abra a Arduino IDE.

Copie e cole o seguinte código na Arduino IDE:  
cpp  
Copiar código  
// Código Arduino

const int ledPin = 13; // LED conectado ao pino digital 13

void setup() {

pinMode(ledPin, OUTPUT); // Define o pino do LED como saída

Serial.begin(9600); // Inicia a comunicação serial a 9600 baud

}

void loop() {

if (Serial.available() > 0) {

char command = Serial.read(); // Lê o comando do serial

if (command == '1') {

digitalWrite(ledPin, HIGH); // Liga o LED

} else if (command == '0') {

digitalWrite(ledPin, LOW); // Desliga o LED

}

}

}

1. Selecione a porta correta e o modelo do seu Arduino no menu "Tools".
2. Clique em "Upload" para carregar o código no Arduino.

### **Passo 3: Configurar o Ambiente Java**

1. **Instale o JDK:**
   * Baixe e instale o JDK a partir do site oficial da Oracle: [JDK Download](https://www.oracle.com/java/technologies/javase-downloads.html).
2. **Configure a Biblioteca RXTX:**
   * Baixe a biblioteca RXTX aqui.
   * Extraia o conteúdo do arquivo baixado.
   * Copie os arquivos .jar para o diretório libs do seu projeto Java.
   * Adicione os arquivos .dll (Windows) ou .so (Linux) ao diretório de bibliotecas do seu sistema.

### **Passo 4: Escrever o Código Java**

1. Crie um novo projeto Java em sua IDE preferida (Eclipse, IntelliJ, NetBeans).
2. Adicione a biblioteca RXTX ao classpath do seu projeto.

Copie e cole o seguinte código Java no seu projeto:  
java  
Copiar código  
import gnu.io.CommPortIdentifier;

import gnu.io.SerialPort;

import java.io.OutputStream;

import java.util.Enumeration;

public class ArduinoControl {

private SerialPort serialPort;

private OutputStream output;

private static final String PORT\_NAME = "COM3"; // Porta COM utilizada pelo Arduino

private static final int TIME\_OUT = 2000;

private static final int DATA\_RATE = 9600;

public void initialize() {

CommPortIdentifier portId = null;

Enumeration portEnum = CommPortIdentifier.getPortIdentifiers();

// Encontra a porta COM especificada

while (portEnum.hasMoreElements()) {

CommPortIdentifier currPortId = (CommPortIdentifier) portEnum.nextElement();

if (currPortId.getName().equals(PORT\_NAME)) {

portId = currPortId;

break;

}

}

if (portId == null) {

System.out.println("Porta COM não encontrada.");

return;

}

try {

// Abre a porta serial e configura os parâmetros

serialPort = (SerialPort) portId.open(this.getClass().getName(), TIME\_OUT);

serialPort.setSerialPortParams(DATA\_RATE, SerialPort.DATABITS\_8, SerialPort.STOPBITS\_1, SerialPort.PARITY\_NONE);

output = serialPort.getOutputStream();

} catch (Exception e) {

System.err.println(e.toString());

}

}

public void close() {

if (serialPort != null) {

serialPort.close();

}

}

public void sendCommand(char command) {

try {

output.write(command);

} catch (Exception e) {

System.err.println(e.toString());

}

}

public static void main(String[] args) {

ArduinoControl arduino = new ArduinoControl();

arduino.initialize();

try {

// Liga o LED

arduino.sendCommand('1');

Thread.sleep(2000); // Espera 2 segundos

// Desliga o LED

arduino.sendCommand('0');

Thread.sleep(2000); // Espera 2 segundos

} catch (InterruptedException e) {

e.printStackTrace();

}

arduino.close();

}

}

1. Certifique-se de que a PORT\_NAME está configurada corretamente para a porta COM usada pelo seu Arduino (verifique no Gerenciador de Dispositivos).

### **Passo 5: Executar o Código Java**

1. Compile e execute o código Java.
2. Observe que o LED no Arduino ligará por 2 segundos e depois desligará por 2 segundos.

### **Solução de Problemas**

* **Porta COM não encontrada:**
  + Verifique se a porta COM está correta.
  + Certifique-se de que o Arduino está conectado corretamente.
* **Erro de comunicação serial:**
  + Verifique se os drivers necessários para a comunicação serial estão instalados.
  + Assegure-se de que a biblioteca RXTX está corretamente configurada.

### **Conclusão**

Seguindo esses passos, você deverá ser capaz de controlar um LED conectado ao Arduino usando um programa Java. Esse exemplo pode ser expandido para controlar diversos dispositivos e sensores, permitindo a criação de projetos mais complexos e interativos.