UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

GUSTAVO ALEXANDRE MACHADO URZUA

Interface gráfica para comando de iluminação de grandes ambientes usando telerruptores

São Carlos
2011

Gustavo Alexandre Machado Urzua

Interface gráfica para comando de iluminação de grandes ambientes utilizando telerruptores

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos, da Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase eletrônica

ORIENTADOR: Prof. Dr. Roberto Clarete Pessotta

São Carlos

2011

Dedico esta conquista aos meus pais que sempre me apoiaram e se dedicaram para que eu pudesse alcançar meus objetivos

Agradecimentos

Gostaria de agradecer aos meus pais, que sempre me apoiaram e confiaram em mim

Aos meus irmãos, Leonardo e Flávia, pela amizade e ajuda que me deram durante toda a minha vida

Aos meus amigos da faculdade, pelo companheirismo na hora dos estudos e também das festas

Aos colegas de republica, que foram praticamente a minha família durante a faculdade, compartilhando momentos de felicidades e de dificuldades

Aos meus amigos Moller e Mussatto, que me ajudaram quando tive dificuldades com a programação em Java

E a todos os funcionários e professores da USP São Carlos, que contribuem para a formação de engenheiros e profissionais de qualidade

Resumo

URZUA, G. A. M. Interface gráfica para comando de iluminação de grandes ambientes utilizando telerruptores. 2011. 40p. Dissertação (Graduação) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2011.

Este trabalho visa a construção de um software para controlar a iluminação do anfiteatro do campus 2 da USP de São Carlos, bem como um painel sinótico para obter mais comodidade e economia. O programa feito pode ser dividido em duas partes: a interface gráfica e a comunicação serial. A interface gráfica estabelecerá a comunicação serial, onde o processador que receberá os comandos do usuário, irá processá-los e enviá-los a um outro processador, que irá fazer a multiplexação do sinal e então executar a operação de liga ou de desliga, através de um circuito eletrônico e de telerruptores, e enviar um sinal de volta para apresentar na interface gráfica o estado das várias cargas comandadas. Este trabalho foi baseado no equilíbrio entre simplicidade e eficiência, buscando a formulação de um algoritmo simples e intuitivo que apresente boas respostas ao que o usuário deseja.

Palavras chaves: domótica, edifícios inteligentes, telerruptores, relé de impulso, display de matriz ativa, interface gráfica.

Abstract

URZUA, G. A. M. Graphical interface for lighting control of large environment using telerruptors. 2011. 40 p. Dissertação (Graduação) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2011.

This work aims to build a software to control the lighting of the amphitheater 2 of USP campus São Carlos, as well as a synoptic panel for more convenience and economy. The program can be done in two parts: the graphical interface and serial communications. The graphical interface will establish the serial communications, where the processor receives user commands, will processa then and send them to another processor that will done multiplexig of the signal and then perform the operation on or off by na electronic circuit and telrruptors, and send a signal back to the graphical display of the state led several charges. This work was based on a balance between simplicity and efficiency, seeking to fomulate a simple and intuitive algorithm to provide good answers to what the user wants.

Keywords: home automation, intelligent buildings, telerruptors, impulse relay, active matrix displays, graphical user interface.

Lista de figuras

Lista de Siglas

ABNT	Associação Brasileira de Normas Técnicas
API	Application Programming Interface
CTS	Clear To Send
DCE	Data Circuit-terminating Equipament
DTE	Data Terminal Equipament
EIA	Electronic Industries Association
GUI	Graphical User Interface
HDLC	High-Level Data Link Control
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
JVM	Java Virtual Machine
LED	Light Emitting Diode
LSB	Least Significant Bit
NBR	Norma Brasileira
PHP	Personal Home Page
RS	Recommended Standard
RTS	Ready To Send
SDRAM	Synchronous Dynamic Random Access Memory
SO	Sistema Operacional
USB	Universal Serial Bus
USP	Universidade de São Paulo
XML	Extensible Markup Language

Sumário

Capítulo 1

Introdução

Na busca de maior controle de iluminação e economia de energia, foi feito um projeto no Anfiteatro do Campus II da USP de São Carlos utilizando telerruptores. Nesse projeto, foi instalado um telerruptor para controlar o acendimento de cada lâmpada do anfiteatro. Por conta disso, é possível comandar o acendimento individual de cada uma das 55 lâmpadas, porém é necessário uma chave do tipo liga/desliga para cada uma delas, o que tornou o quadro de instrumento muito grande. Além disso, foi feito um comando para cada linha e coluna, bem como de algumas outras cenas, que fez aumentar mais ainda o tamanho do quadro de comando e a complexidade da fiação elétrica. O método empregado dificulta também a mudança ou a adição de outro comando para um novo conjunto de lâmpadas.

Para resolver esse problema, foi proposto a criação de uma interface gráfica para comandar a iluminação. A interface irá substituir os interruptores por botões virtuais. Os botões passam a ter a dupla função de liga e desliga, além de mudarem a aparência da figura representativa da luminária para saber em qual estado está a lâmpada. Também, é possível criar até 18 grupos que são chamados através de menus, que podem ser modificados e renomeados de forma simples e pratica, permitindo ao usuário criar as cenas que ele deseja. Sendo assim, economiza-se a parte de “hardware”, composta pelos interruptores e pela fiação elétrica, além de aumentar a flexibilidade do controle da iluminação.

O projeto será desenvolvido em duas partes, por dois alunos da graduação da Engenharia Elétrica. A primeira parte será feita neste momento, e envolverá a construção da interface gráfica, a escolha do microcontrolador que irá rodar esta interface e será o processador mestre, e a escolha da comunicação que será usada entre os dois microcontrolador. Já a segunda parte será feita pelo aluno Cesar Maia, e ficará responsável em fazer a multiplexação do sinal recebido do microcontrolador que controla a interface gráfica. O processador que faz a multiplexação irá executar a operação de liga e desliga e enviar um sinal via comunicação RS232 para o microcontrolador da interface gráfica que irá mostrar o estado das varias cargas comandadas. Está operação de liga e desliga será executada através de um circuito eletrônico e telerruptores.

Neste Capitulo 1, iremos abordar os assuntos referentes a interface gráfica, do ambiente de desenvolvimento, da linguagem de programação, dos telerruptores e do microcontrolador escolhido.

1.1 Interface Gráfica do Usuário

A interface gráfica do usuário (GUI – *Graphical User Interface*) fornece a um programa uma aparência e um comportamento diferenciado, ou seja, apresenta uma interface pictórica para um programa. Por meio de um conjunto consistente de componentes intuitivos de interface com o usuário, fornecidos por diferentes aplicativos, as GUI familiarizam o usuário com um programa mesmo quando ele nunca o usou. Portanto, a interface gráfica ajuda o usuário a aprender a usar o programa em um tempo bem menor e também, aumenta a habilidade de usar o programa de forma mais produtiva (DEITEL, 2003).

Para se construir uma GUI, é necessário componente GUI, também chamado de controle ou widget. Esses componentes são objetos capazes de ser interagidos pelo usuário por meio de um mouse, teclado, telas sensíveis ao toque ou outra forma de entrada, como o reconhecimento de voz.

Programas com interface gráfica são “eventos dirigidos”. Isto significa que o programa reage a ações do usuário, chamadas de eventos. Alguns exemplos de eventos são pressionar uma tecla, mover o mouse, pressionar um botão ou selecionar um item do menu (FISHER, 2005).

As primeiras interfaces gráficas foram criadas na década de 70 e tiveram uma rápida expansão em meados de 80, ajudando em muito a popularização dos computadores devido a não necessidade de ter um treinamento formal para utilizá-lo (RAYMOND, 2004).

Figura 1.1: Exemplo de interface gráfica do usuário

Devido às facilidades trazidas pela interface gráfica, sua utilização passou a ser muito comum em diversos dispositivos eletrônicos, muito deles usados na automatização de processos, onde o usuário pode ter maior facilidade de compreensão do sistema, garantindo maior controle e segurança.

Pensando nisso, aglutinamos as vantagens da interface gráfica com a simplicidade dos relés de impulsos para construir sistemas de controle de iluminação visando maior comodidade e economia de energia.

1.2 Telerruptores

Um exemplo de telerruptor é o relé de impulso. Esses relés são dispositivos eletromecânicos que permitem o acionamento de mais de um cenário de iluminação a partir de um mesmo pulsador simples. Ao ser inserido no controle de um sistema de iluminação, o relé de impulso tem o objetivo de alterar o estado ou posição dos contatos quando em sua bobina é aplicada uma tensão através de um pulso mínimo de 100ms. No instante do pulso, o efeito eletromagnético faz com que uma alavanca movimente um pequeno Came (tipo de roda dentada) que abre ou fecha os contatos. Ele pode ser utilizado como alternativa para ligações de interruptores paralelos e intermediários, trazendo economia no dimensionamento de condutores, pois o circuito de comando passa a ser executado com condutores de bitola mínima de 0,5mm², de acordo com o que prescreve a Norma NBR5410 (Instalações Elétricas de Baixa Tensão).

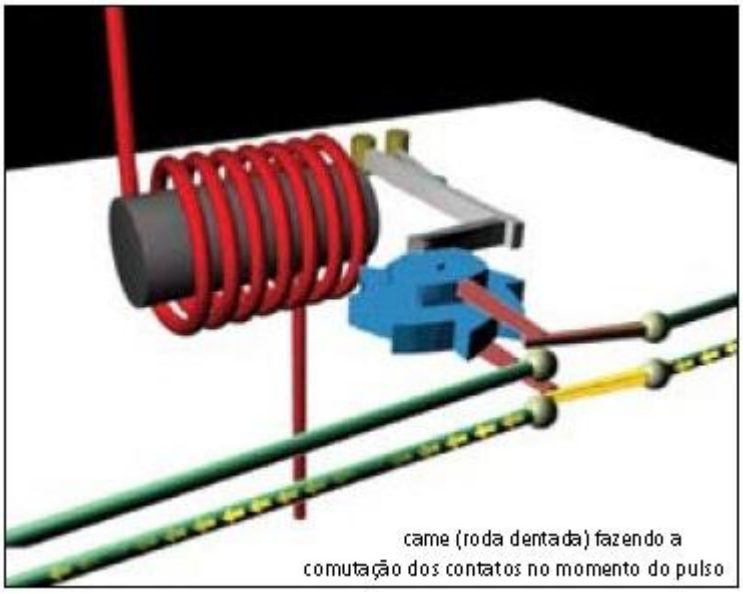

Figura 1.2: Figura do came fazendo a comutação dos contatos

Há relés de impulso disponíveis para diferentes tipos de montagem, valores de tensão e sequencias de acionamento. Além do acionamento por pulsadores, há a possibilidade da instalação de um sistema de controle remoto paralelo ao pulsador.

Figura 1.3: Relês de impulso

Por utilizar a fiação de comando separada da fiação de potencia, que tem bitola maior (no mínimo 1,5mm² de acordo com a mesma norma NBR5410), é possível usar o fio de comando também para receber a sinalização do estado da carga, ou seja, se ela está ligada ou desligada. Em particular, utiluzaremos um relé de impulso que agrega um circuito eletrônico, ou seja, não é um dispositivo puramente eletromecânico como os apresentados nas figuras acima. Então, podemos utilizar meio ciclo de uma senoide para enviar o comando e a outro meio ciclo para receber para receber a informação do estado da carga.

1.3 NetBeans IDE 6.9.1.

O NetBeans é um ambiente de desenvolvimento integrado (IDE) disponível para Windows, Mac, Linux, e Solaris. Foi criada no final da década de 90 pela Sun Microsystem, pertencente a Oracle desde 2010, e é um dos mais antigos projetos livres de ambientes de desenvolvimento integrado (MECENAS, 2008). Consiste em

um ambiente de desenvolvimento de código aberto e uma plataforma de aplicação que permite desenvolvedores criar rapidamente aplicação para web, empresas, desktop e dispositivos moveis usando a plataforma Java, bem como PHP, JavaScript, Ajax, Groovy, Grails e C/C++. Ele é executado em varias plataformas como Windows, Linux, Solaris e MacOS.

A IDE NetBeans auxilia programadores a escrever, compilar, debugar e instalar aplicações, e foi arquitetada em forma de uma estrutura reutilizável que visa simplificar o desenvolvimento e aumentar a produtividade, pois reúne em uma única aplicação todas estas funcionalidades. Foi totalmente escrito em Java, porem suporta qualquer linguagem de programação que desenvolva com Swing, alem de suportar linguagens de marcação como XML e HTML.

Essa IDE fornece uma base solida para a criação de projetos e módulos, pois possui um grande conjunto de bibliotecas, módulos e APIs (do inglês *Application Program Interface*) além de uma documentação vasta e bem organizada. Tais recursos auxiliam o desenvolvedor a escrever seu software de maneira mais rápida. Como o NetBeans é escrito em Java, é independente de plataforma, funciona em qualquer sistema operacional que suporte a maquina virtual Java (JVM)

1.4 Java

Java é uma linguagem de programação orientada a objeto, baseada em C++, desenvolvida na década de 90 pela empresa Sun Microsystems. Linguagem orientada a objetos implementa um conjunto de classes que definem os objetos presentes no sistema de software. Cada classe determina o comportamento (definido nos métodos) e estados possíveis (atributos) de seus objetos, assim como o relacionamento com outros objetos. Os objetos de uma classe têm a capacidade de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objetos. Diferentemente das linguagens convencionais, que são compiladas para o código nativo, a linguagem Java é compilada para um *bytecode* que é executado por uma maquina virtual (DEITEL, 2003).

Em Caelum (2004), temos a seguinte situação quando vamos compilar um programa em uma linguagem como C ou Pascal:

Figura 1.4: Funcionamento do compilador C

O código fonte é compilado para o código de maquina especifico de uma plataforma e sistema operacional (SO). Este sistema operacional executa o código resultante e, por esse motivo, tem um código executável para cada sistema operacional.

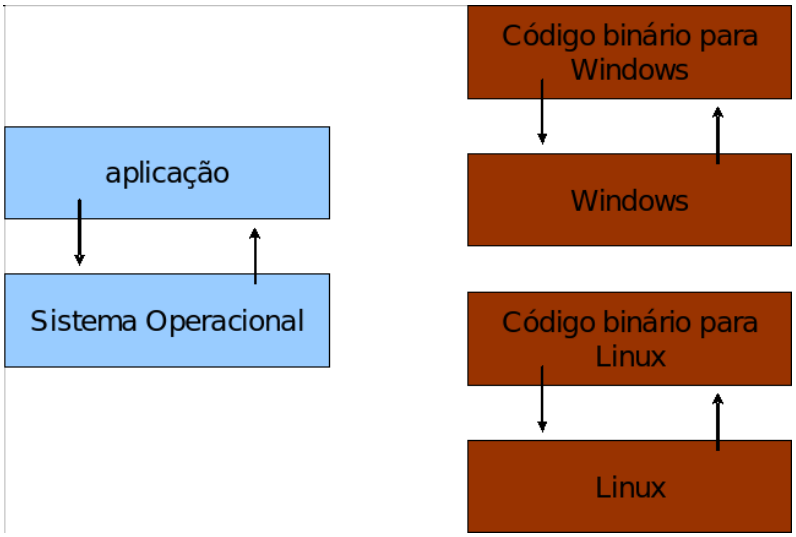

Figura 1.5: Geração do código binário para diferentes Sistemas Operacionais

Para se utilizar o software em varias plataformas, é necessário compilar uma vez para cada sistema operacional, no qual irá rodar o programa. A sua aplicação se utiliza das bibliotecas do sistema operacional, como, por exemplo, a de interface gráfica para desenhar as telas que deferem de plataforma para plataforma. Já o Java utiliza do conceito de maquina virtual, onde existe, entre o sistema operacional e a aplicação, uma camada extra responsável por “traduzir” o que sua aplicação deseja fazer para as respectivas chamadas do sistema operacional onde ela está rodando no momento:

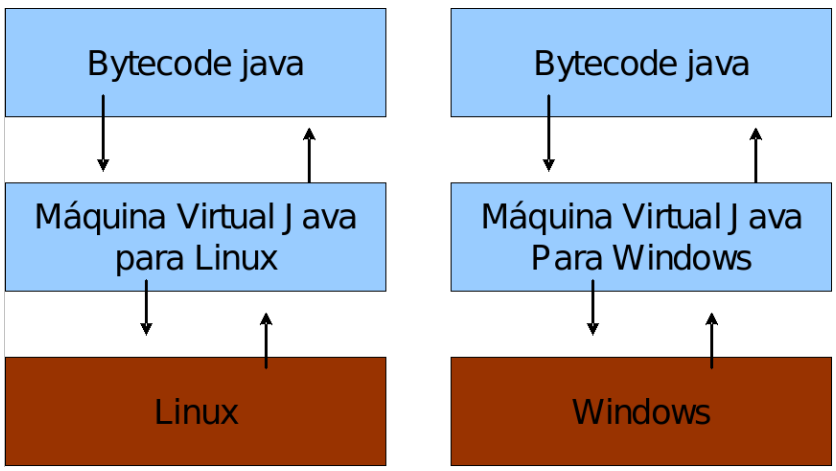

Figura 1.6: Funcionamento da Máquina Virtual Java para Linux e Windows

Dessa forma, a maneira com a qual abre uma janela do Linux ou no Windows é a mesma, ou seja, não depende do sistema operacional. A maquina virtual não entende código Java, ela entende um código de maquina especifico gerado por um compilador Java, como o javac, e é conhecido por "*bytecode*", pois existem menos de 256 codigos de operação dessa linguagem, e cada "opcode" gasta um byte. O compilador Java gera esse *bytecode* que, diferente das linguagens sem maquina virtual, vai servir para diferentes sistemas operacionais.

A linguagem de programação Java é a linguagem convencional da plataforma Java, mas não é a única linguagem.

Desde seu lançamento, a plataforma Java foi adotada mais rapidamente do qualquer outra linguagem de programação na historia da computação. Atualmente Java é uma referencia no mercado de desenvolvimento de software, com um vasto ambiente de execução como navegadores, mainframes, sistemas operacionais, celulares, etc (DEITEL, 2003).

1.5 Comunicação serial RS 232

RS 232 (do inglês *Recommended Standard 232*) é uma padronização criada na década de 60 para a troca serial de dados binário entre um DTE (*Data Terminal Equipment*) e um DEC (*Data Circuit-terminating Equipment*) (ANDRÉ, 2002). O padrão RS 232, criado pela EIA (*Eletronics Industries Association*), surgiu para facilitar a comunicação, pois antigamente a comunicação era feita através de linhas telefônicas que conectava um computador central a outros computadores. Por conta disso, o padrão define as características elétricas e sincronismo de sinais, o significado dos sinais, o tamanho físico e pinagem de conectores.

Em Strangio (2006), no protocolo de comunicação serial RS 232, caracteres são enviados um a um como um conjunto de bits, normalmente começando a sequência com o bit menos significativo (LSB). A codificação mais comumente usada é o "*start-stop* assíncrono" que usa um bit de inicio, seguido por sete ou oito bits de dados, possivelmente um bit de paridade, e um ou dois bits de paragem sendo então necessário pelo menos 10 bits para enviar um único caractere. Tal fato acarreta a necessidade de dividir por um fator de dez a taxa de transmissão para obter a velocidade de transmissão. A alternativa mais comum ao "*start-stop* assíncrono" é o HDLC (do inglês, *High-level Data Link Control*). O padrão define os níveis elétricos correspondentes aos níveis lógicos um e zero, a velocidade de transmissão padrão e

os tipos de conectores. Por se tratar de uma comunicação assíncrona, cabe ao transmissor e ao receptor efetuarem o controle de tempo para saber o inicio e o fim de cada bit.

Para o controle de fluxo via hardware, o padrão RS 232 normalmente usa dois sinais: RTS (*Ready To Send*) e o CTS (*Clear To Send*). Para saber quando o transmissor vai iniciar o envio, ele sinaliza através do pino RTS. Quando o receptor capta o sinal, ele se prepara para receber o dado, depois que está pronto, sinaliza através do pino CTS. Depois que os dois pinos estão setados inicia-se a transmissão. Conhecendo a velocidade de transmissão, ou seja, a quantidade de bits por segundo transmitida, e o bit de inicio, basta o receptor esperar o bit de parada e finalizar a transmissão, depois ficar aguardando o próximo bit de inicio para começar uma nova transmissão

As velocidades de transmissão comuns são: 300, 1200, 2400, 9600, 19200 bits por segundo. Tipicamente ambos os dispositivos devem estar configurados com a mesma velocidade. A paridade é um método de verificar a precisão dos dados, porém normalmente é nula, ou seja, não é usada. Quando usada ela pode ser par ou impar, acomodando os dados para a contagem de bits 1 ser par ou impar, podendo assim o receptor detectar a transmissão de erros. Os bits de parada são enviados no fim de cada byte transmitido com o intuito de permitir que o receptor do sinal se sincronize.

Outras configurações definem quando pinos enviam sinais de “*handshake*”, ou outras checagens de integridade dos dados. Caracteres no fluxo de dados indicam para o receptor que o remetente do caractere está pronto para receber mais dados ou se deve parar de enviar dados.

O RS 232 é recomendado para conexões curtas. Os sinais variam de 3 a 15 volts positivos ou negativos, valores próximos de zero não são sinais validos. O nível lógico um é definido por ser voltagem negativa e tem significado funcional de OFF (desligado). O nível lógico zero é positivo e a função é ON (ligado). Abaixo há um exemplo de como é enviado um caractere:

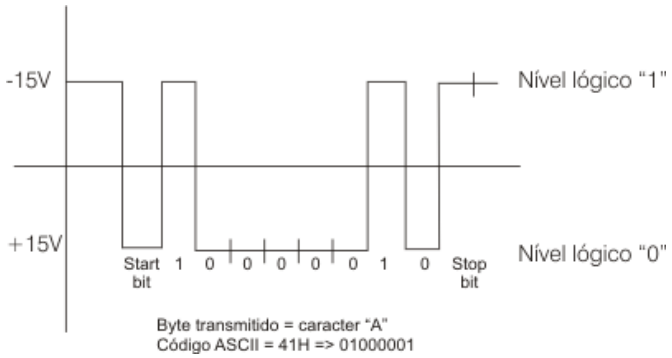

Figura 1.7: Transmissão de um byte

Há dois tipos de conectores mais usados no padrão RS 232: DB9 e DB25. Os números dizem a quantidade de pinos que tem cada conector. Abaixo temos a figura desses conectores:

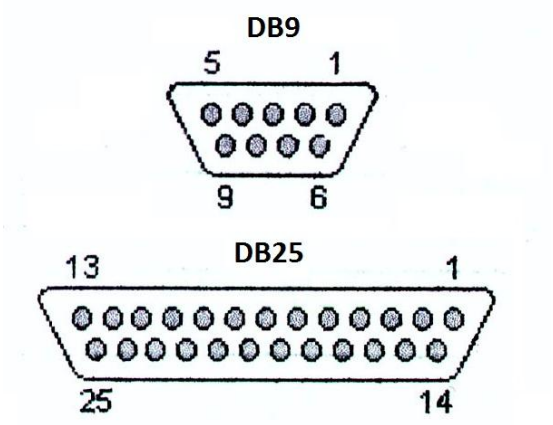

Figura 1.8: Portas RS232 DB9 e DB25.

Abaixo temos a pinagem dos dois conectores e a função de cada pino:

DB25	DB9	SINAL	DESCRIÇÃO	FUNÇÃO
2	3	TX	Transmit Data	O
3	2	RX	Receive Data	I
4	7	RTS	Request to Send	O
5	8	CTS	Clear to Send	I
6	6	DSR	Data Set Ready	I
20	4	DTR	Data Terminal Ready	O
8	1	DCD	Data Carrier Detect	I
22	9	RI	Ring Indicator	I
7	5	GND	Signal Ground	GND
1	-----	GND	Signal Ground	GND

Figura 1.9: Tabela descritiva de cada sinal das portas DB25 e DB9

Na linguagem Java, é necessário uma API (do inglês *Application Programming Interface*), que é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por programas aplicativos que não entram em detalhes da implementação do software, mas apenas usar seus recursos, para executar a comunicação serial RS232. A API utilizada foi a RXTXSerial-2.1-7-bins, um arquivo executável Java, da empresa Sun, que fornece a API totalmente de graça pelo site www.rxtx.org.. A RXTX Serial é portável para Linux, Windows e Mac, trazendo todas as bibliotecas e funções necessárias para a comunicação do software desenvolvido com as portas seriais do controlador.

1.6 O Kit SAM9-L9260

O kit escolhido para realizar o processamento da interface gráfica e a comunicação serial foi o SAM9-L9260 da Olimex. Esse kit é uma plataforma de desenvolvimento de baixo custo com microcontrolador ARM9, 64MB de SDRAM e 512MB de memória flash NAND. A placa tem conector Ethernet 100Mbit, servidor USB, dispositivo USB, porta serial RS232 e uma porta com 40 pinos com todas as portas do SAM9260 não usadas disponíveis para adicionar. Esse kit tem uma boa quantidade de memória Flash e Ram e roda Linux, WindowsCE e outros sistemas operacionais. Abaixo estão as características do kit:

Processador: AT91SAM9260 16/32 bit ARM9 200MHz
Clock de 50MHz
Conector JTAG padrão com ARM 2x10 de pinagem para programação/ debugging com AMR-JTAG
64MB de SDRAM

512MB de memória Flash NAND

Conector Ethernet de 100Mbit

Conectores USB

Drivers e interface RS232

Conector de cartão SD/MMC

Um botão do usuário e um botão de reset

Uma LED para fonte e dois LEDs de status

Um regulador de tensao on board de 3,3V com capacidade de até 800mA

Fonte de alimentação de 5V DC

Capacitor para filtragem na fonte de alimentação

Cristal de 18,432MHz

Porta de extensão para comunicação

Dimensões: 100 x 80mm

Figura 1.10: Kit SAM9-L9260

O motivo de ter escolhido esse kit foi o fato de ter comunicação RS232 e Linux Debian 5.0, o que possibilitou instalar a linguagem Java nele. Para usá-lo, precisou conectar um cabo null-modem do tipo fêmea-fêmea na placa e na porta serial do computador. Usando o HyperTerminal do Windows e configurando a baud rate em 115200, 8 bits de dados, 1 bit de parada e nenhum bit de paridade foi possível usar o Linux que está pré-carregado nas memórias flash NAND e DATAFLASH e fazer o teste da interface gráfica, bem como de sua comunicação serial RS232.

Capítulo 2

A programação

Para dar inicio ao projeto, precisou-se de uma familiarização com o ambiente de desenvolvimento Netbeans, bem como a linguagem de programação Java. Dentro do ambiente de programação foi criado um novo projeto, com o nome de Anfiteatro. Todas as partes relacionadas com a interface gráfica e com a comunicação serial estão contidas nesse projeto. O projeto é um aplicativo Java SE padrão que utiliza um script de construção Ant gerado pelo IDE para construir, executar e depurar o projeto. Foram criados dois pacotes dentro do projeto: Pacote anfiteatro e pacote serial. No pacote anfiteatro foram criadas duas classes (Main e RecebeDados), dois Frames (JFAnf e Lâmpadas), que são componentes principais das aplicações gráficas, onde podem conter elementos de interação com o usuário, além de uma barra com título e botões para redimensionar, minimizar, maximizar e fechar (FISHER, 2005). As figuras usadas na interface gráfica também estão dentro desse pacote. Já no pacote serial foram criadas duas classes (SerialCom e SerialComLeitura) que são responsáveis pela comunicação serial.

1.1 JFAnf

O JFAnf é o frame principal da parte gráfica, ou seja, é onde foi feita a parte gráfica do anfiteatro. Primeiramente foi colocado um painel dentro do frame que alocou os botões que simulavam as lâmpadas. Começou-se a construir a parte gráfica com as mesmas características, em termos de disposições das lâmpadas, encontradas no anfiteatro do campus dois. As 55 lâmpadas foram postas e nomeadas, para mais fácil entendimento, de forma matricial, com exceção das lâmpadas do palco, que não pertenciam a nenhuma coluna. Também foi posta a lâmpada que fica responsável de iluminar o quadro de comandos e nomeada de IQC (Iluminação do Quadro de Comando).

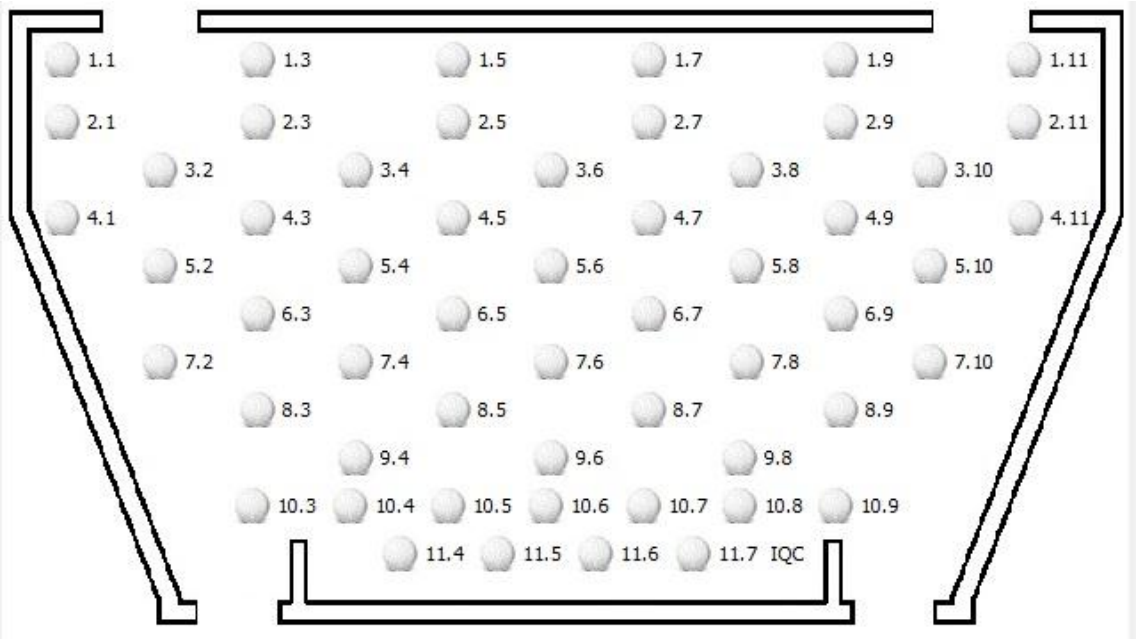

Figura 2.1: Interface gráfica do JFAnf

Cada botão segue a lógica de alternar o estado cada vez que ele é clicado, além de mudar o ícone quando muda de estado. Além disso, irá enviar um byte com as informações do estado e sua posição via comunicação serial RS 232. Como todas as lâmpadas seguem a mesma lógica, foi criado um método que foi aplicado em cada uma delas.

```
        private void AlternarEstado(ActionEvent event) {
                javax.swing.ImageIcon icone = (ImageIcon)
((JButton)event.getSource()).getIcon();
                 if (icone.getDescription() == null || icone.getDescription().equals("acesa")) {
                    icone = new
javax.swing.ImageIcon(getClass().getResource("apagada.jpg"));
                    icone.setDescription("apagada");
                    ((JButton)event.getSource()).setIcon(icone);
                 } else {
                    icone = new
javax.swing.ImageIcon(getClass().getResource("acesa.jpg"));
                    icone.setDescription("acesa");
                    ((JButton)event.getSource()).setIcon(icone);}
```

```
}
```

Esse método verifica por comparação qual o estado da lâmpada, e usando uma lógica “if” ele envia o sinal para mudança de estado e depois de ter recebido a confirmação que não ocorreu nenhum erro na comunicação, ele muda o ícone do botão.

Além dos botões que representam as lâmpadas, foram criados mais três botões para criar os grupos, ligá-los e desligá-los. Os três botões chamam um pop-up menu com dezoitos menus, nomeado de “Novo Grupo”. Os menus chamados pelo botão “Criar Grupos” tem a função de chamar uma janela onde é possível colocar o nome do grupo e as lâmpadas que pertencem a esse grupo. Para se chamar essa janela dentro do Frame JFAnf, foi criado um objeto do frame Lâmpadas que será declarado como uma “variavel”. Os botões “Ligar” e “Desligar” vão chamar o menu com os grupos criados e vai executar a ação solicitada pelo usuário.

O método usado para saber quais são as lâmpadas de cada grupo também foi por comparação através da lógica “if”.

```
if (objeto.matriz[1][1] == 1) {
        javax.swing.ImageIcon icone = (ImageIcon) jB11.getIcon();
        icone = new
javax.swing.ImageIcon(getClass().getResource(status+".jpg"));
        icone.setDescription(status);
        jB11.setIcon(icone);
            }
```

Ele compara cada elemento da matriz e vê se ele está setado, ou seja, se ele recebe 1. Caso seja positivo, ele executa a ação, senão, testa o próximo elemento da matriz.

2.2 Lâmpadas

O frame Lâmpadas tem a função de gerar a parte gráfica da janela que abre quando um menu do “Criar Grupos” é acionado.

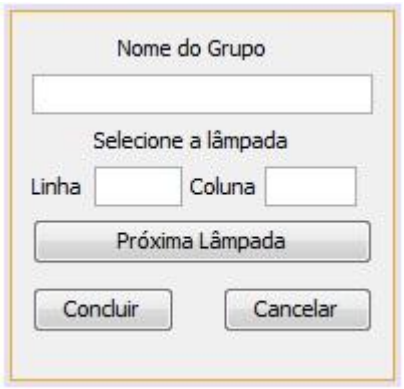

Figura 2.2: Interface gráfica da janela Lâmpadas

Os dados inseridos no campo “Nome do Grupo” são armazenados na forma de “String” e colocados em um vetor criado na classe principal. Já nos campos “Linha” e “Coluna” os dados são convertidos para inteiro e depois são guardados como matriz, setando o elemento correspondente, que depois será lido quando a lógica dos menus criados pelos botões “Ligar” e “Desligar” do JFAnf é executada. Ao se clicar no botão “Concluir” ou “Cancelar” essa janela fecha e os dados nela fornecidos são armazenados.

2.3 RecebeDados

A classe RecebeDados tem a única função de criar a String nome, posição e a matriz de inteiros que serão usados no vetor criado na classe principal.

```
public class RecebeDados {
   public String nome;
   public String pos;
   public int[][] matriz = new int [13][13];
}
```

2.4 Classe Principal

A classe principal, chamada de Main, tem a função de gerenciar e executar o programa. Logo no inicio dela, as classes “SerialCom” e “SerialComLeitura”, que pertencem ao pacote “Serial”, são importadas. Além disso, como foi feito com o frame “Lâmpada” no algoritmo do frame “JFAnf”, o frame “JFAnf” é declarado como objeto e chamado logo que roda o programa. Dentro da classe Main, foi criado um vetor, da biblioteca Vector, capaz de armazenar 100 posições. Esse vetor, foi a solução encontrada, para guardar as informações do frame “Lâmpadas” e lido posteriormente no frame “JFAnf”. Para isso, foram criadas as variáveis da classe “RecebeDados” que irão definir a posição e os dados como o nome e as lâmpadas do grupo que irão ser colocados nessa posição. Também é na classe principal que a comunicação serial é rodada, definindo a porta com que será feita a comunicação, a velocidade de transmissão, ou Baud Rate, e o bit de paridade.

```
package anfiteatro;
import serial.SerialCom;
import serial.SerialComLeitura;
import java.util.Vector;
/**
* @author Gustavo
 */
public class Main extends SerialCom {
    public static Vector lampada = new Vector(100);
     /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        RecebeDados name = new RecebeDados();
        name.nome = "Novo Grupo";
        lampada.add(name);
        lampada.add(name);
        lampada.add(name);
        lampada.add(name);
        lampada.add(name);
        lampada.add(name);
        lampada.add(name);
```

```
    lampada.add(name);
    lampada.add(name);
    lampada.add(name);
    lampada.add(name);
    lampada.add(name);
    lampada.add(name);
    lampada.add(name);
    lampada.add(name);
    lampada.add(name);
    lampada.add(name);
    lampada.add(name);
    lampada.add(name);
    lampada.add(name);
    lampada.add(name);
    lampada.add(name);
    lampada.add(name);

   /*  SerialComLeitura leitura = new SerialComLeitura("COM11", 9600, 0);
    leitura.HabilitarLeitura();
    leitura.ObterIdDaPorta();
    leitura.AbrirPorta();
    leitura.LerDados();
    //Controle de tempo da leitura aberta na serial
    try {
        Thread.sleep(10000);
    } catch (InterruptedException ex) {
        System.out.println("Erro na Thread: " + ex);
    }
    leitura.FecharCom();

SerialComLeitura serialEscrita = new SerialComLeitura("COM11", 9600, 0);
serialEscrita.HabilitarEscrita();
serialEscrita.ObterIdDaPorta();
serialEscrita.AbrirPorta();
serialEscrita.EnviarUmaString("Byte enviado");
serialEscrita.FecharCom(); */
    JFAnf anf = new JFAnf();
    anf.setVisible(true);
```

```
        }
    }
```

2.5 Classe SerialCom

A classe SerialCom será responsável por buscar e identificar as portas seriais do dispositivo. Já a classe SerialComLeitura irá fazer a leitura, ou seja, receber as informações e a escrita, enviar informações.

Para que o compilador encontre as bibliotecas necessárias, a API foi instalada devidamente na pasta do IDE Netbeans e importada no inicio do programa da seguinte forma:

```
import gnu.io.CommPortIdentifier;
import java.util.Enumeration;
```

Agora já é possível encontrar as portas seriais do microcontrolador e para isso, os seguintes métodos para obter as portas e verificar se a porta existe foram criados:

```
public String[] ObterPortas(){
    return portas;
  }

protected void ListarPortas(){
    int i = 0;
    portas = new String[10];
    while (listaDePortas.hasMoreElements()) {
        CommPortIdentifier ips =
        (CommPortIdentifier)listaDePortas.nextElement();
        portas[i] = ips.getName();
        i++;
    }
}

public boolean PortaExiste(String COMp){
    String temp;
    boolean e = false;
```

```
while (listaDePortas.hasMoreElements()) {
    CommPortIdentifier ips = (CommPortIdentifier)listaDePortas.nextElement();
    temp = ips.getName();
    if (temp.equals(COMp)== true) {
    e = true;
}
}
return e;}
```

Com a classe SerialCom feita, as portas seriais já podem ser acessadas.

2.6 Classe SerialComLeitura

A partir disso, a classe SerialComLeitura foi criada com o objetivo de enviar e receber as informações. Para isso, as seguintes bibliotecas foram importadas da API:

```
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
```

Depois de ter importado as bibliotecas, os métodos para habitar escrita e leitura, obter o ID da porta, abrir e fechar a porta, ler dados, habilitar o evento serial e enviar um byte foram criados. Com isso, já é possível fazer a comunicação serial, que será chamada toda vez que inicia o programa e irá trocar dados quando alguma lâmpada ou grupo for pressionado. O método para enviar byte segue abaixo:

```
public void EnviarUmByte(String msg){
    if (Escrita==true) {
        try {
            saida = porta.getOutputStream();
            System.out.println("FLUXO OK!");
```

```
            } catch (Exception e) {
                System.out.println("Erro.STATUS: " + e );
            }
            try {
                System.out.println("Enviando um byte para " + Porta );
                System.out.println("Enviando : " + msg );
                saida.write(msg.getBytes());
                Thread.sleep(100);
                saida.flush();
            } catch (Exception e) {
                System.out.println("Houve um erro durante o envio. ");
                System.out.println("STATUS: " + e );
                System.exit(1);
            }
        } else {
            System.exit(1);
        }
}
```

Capítulo 3

Resultados

Os resultados obtidos com a construção do projeto e testes foram positivos, funcionando dentro do esperado. A seguir, serão detalhados os resultados encontrados da interface gráfica, da comunicação serial RS232 e final.

3.1 Interface Gráfica

A interface foi projetada para seguir fielmente o layout do anfiteatro do campus 2. Com a disposição das lâmpadas e sua marcação de forma matricial, ficou intuitivo o controle da iluminação. Além disso, com a mudança do ícone da lâmpada ligada e desligada fica extremamente fácil saber o estado da lâmpada. Por essa razão, a interface também cumpre o papel de painel sinótico, aumentando mais ainda o controle da iluminação. Com a implementação do menu que permite a criação de grupos de acordo com a vontade do usuário, o programa se mostrou bastante flexível e mais vantajoso que o painel de comando com interruptores mecânicos. Não podemos deixar de citar, que foi possível reduzir em muito o espaço físico do quadro de comando além de passar uma sensação de modernidade. Devemos ressaltar que a linguagem Java e o ambiente de desenvolvimento NetBeans facilitaram bastante a criação da interface.

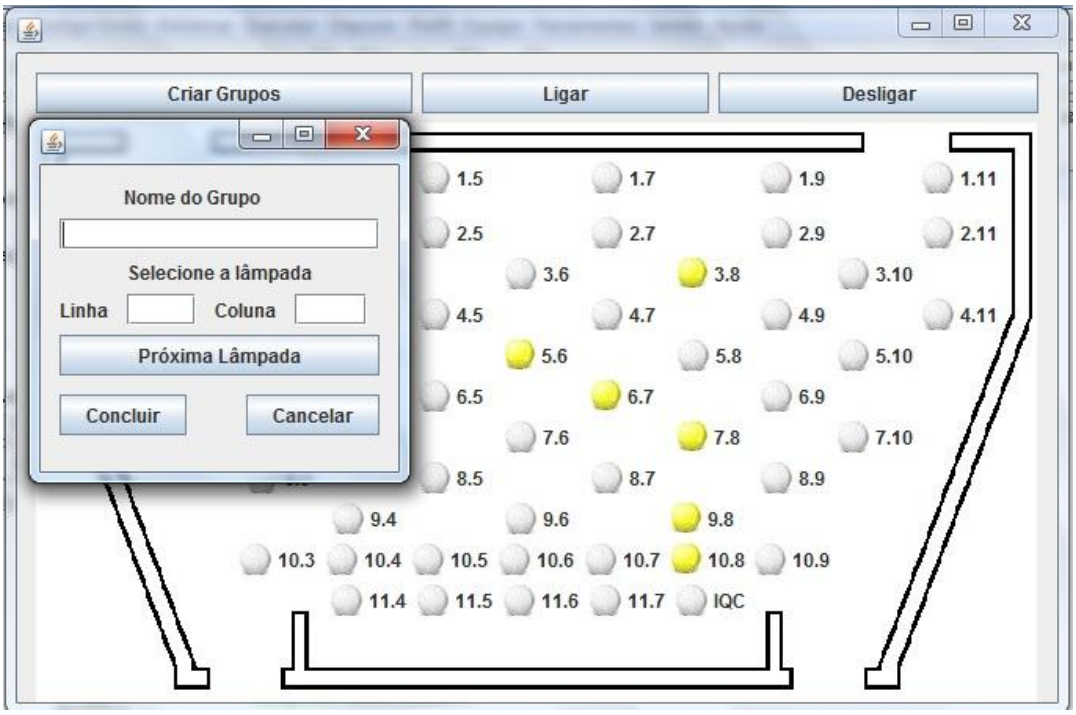

Figura 3.1: Funcionamento da interface gráfica

Na figura acima, é possível perceber a interface em funcionamento, com a janela de configuração de novos grupos. As lâmpadas amarelas representam as lâmpadas acesas e as lâmpadas brancas representam as lâmpadas apagas.

3.2 Comunicação Serial RS232

Para fazer o teste da comunicação serial foi utilizado o programa HyperTerminal do sistema operacional Windows. O HyperTerminal é um aplicativo que permite conectar o computador a outros sistemas remotos. Através da porta RS232, e com os ajustes da baud rate e do bit de paridade, foi possível testar tanto a recepção quanto a transmissão dos dados. Para isso, toda vez que se clicava em uma lâmpada se esperava aparecer na tela do HyperTerminal os bytes enviados, que contem o estado e o endereço da lâmpada. A velocidade de transmissão e recepção está de acordo com o esperado, dando uma resposta quase que instantânea ao comando do usuário.

Figura 3.2: Funcionamento da comunicação serial RS232

3.3 Resultado Final

O resultado final só poderá ser testado depois da segunda parte do projeto, que será a parte da analise e multiplexação do sinal. Depois de feita a multiplexação, o

processador enviara o sinal para um circuito eletrônico que mandará o sinal para o relé de impulso, finalizando o processo de transmissão.

Capítulo 4

Conclusão e trabalhos futuros

Nas conclusões finais, iremos falar das vantagens e desvantagens do projeto, além de fazer uma abordagem de aplicações futuras.

4.1 Conclusão

Os objetivos desse trabalho foram alcançados com sucesso. A escolha da linguagem Java com o ambiente de desenvolvimento Netbeans permitiram a criação de uma interface gráfica que atendesse as necessidades do projeto. As ferramentas Java Swing, como os painéis, botões e menus pré definidos facilitaram na montagem do layout em questão, otimizando em tempo e entendimento, facilitando também a sua mudança, quando necessária. Não podemos esquecer das facilidades para a criação de novas janelas, que deixam a interface com um visual mais limpo e passam a sensação de sofisticação do programa. Apesar da linguagem Java não ter uma biblioteca com as funções necessárias para fazer a comunicação serial RS232, a Interface de Programação de aplicativo (API) RXTX, que é disponibilizada gratuitamente no site do fabricante, facilitou o processo e permitiu que a comunicação entre os dois microcontroladores fosse feita. Devemos lembrar também, que o microcontrolador SAM9-L9260 escolhido tem a flexibilidade de instalação de programas, devido ao fato de ter um sistema operacional pré instalado em sua memória flash. Além disso, já tem porta serial RS232, fundamental para a execução do projeto. Em suma, apesar de ter escolhido uma linguagem que não se tinha conhecimento, as vantagens desta para a criação da interface gráfica compensaram o tempo de aprendizado e familiarização.

4.2 Trabalhos futuros

As conclusões acerca do trabalho desenvolvido mostram que a união de uma interface gráfica ao relé de impulso traz uma serie de vantagens com relação a controle, flexibilidade de criar cenários e economia de espaço físico e de energia.

Trabalhos futuros que permitam maior flexibilidade com relação a criação do layout, bem como o numero de lâmpadas e outros tipos de cargas podem ser implementados em qualquer tipo de ambiente, expandindo mais ainda sua utilização.

Com o avanço da tecnologia, os microcontroladores ficando mais baratos e sua capacidade de processamento ficando maior, a eficiência do projeto ira ficar melhor ainda, podendo, quem sabe um dia, ser uma tecnologia acessível e de fácil implementação tanto para o setor industrial, comercial e residencial.

Referências Bibliográficas

FISHER, P. An Indroduction To Graphical User Interface With Java Swing. First Edition. UK: Addison Wesley, 2005.

ANDRÉ, P. La Liaison RS232. Second Edition. France: Dunod, 2002.

DEITEL, H. M.; DEITEL, P. J. Java Como Programar. Quarta Edição. Brasil: Bookman, 2002.

GONCALVES, E. Dominando o NetBeans. Primeira Edição. Brasil: Moderna, 2006.

MECENAS, I. NetBeans 6.1 Desenvolvendo em Java e Ruby. Primeira Edição. Brasil: Alta Books, 2008.

CAELUM, Java e Orientação a Objetos. Primeira Edição. Brasil: Caelum, 2004.

Sites consultados:

Java. Acedido em: Abril, 2011, em: http://www.java.com/pt_BR/about/

Netbeans. Acedido em: Abril, 2011, em: http://netbeans.org/about/

MAIRINK, F. Instalacoeseletricas. Acedido em: Maio, 2011, em: http://instalacoeseletricas.com/

Interface Central Treasure. Acedido em: Maio, 2011, em: http://interface.centraltreasure.com/

Arner Robotics. Acedido em: Maio, 2011, em: http://www.arnerobotics.com.br/eletronica/comunicacao_rs232_PIC.htm

Manel Parede. Acedido em: Maio, 2011, em: http://manelparedes.blogspot.com/2009/11/interface-grafica-do-utilizador-gui.html