UM SISTEMA DE CONTROLE DE ACESSO UTILIZANDO: APIS JAVACOMM E JDBC, MICROCONTROLADOR PIC E BANCO DE DADOS RELACIONAL

Dalasiel L. dos SANTOS (1); Jardenes F. de OLIVEIRA JUNIOR (1); Jean F. LOPES (1); Katyusco de F. SANTOS (1)
(1) Tecnologia em Sistemas de Informação – Centro Federal de Educação Tecnológica de Pernambuco – CEFETPE
Av. Prof. Luis Freire, 500, CEP 50740-540, Cidade Universitária, Recife-PE, (81) 2125-1600
dalasiel_lima@yahoo.com.br, jardenesjr@yahoo.com.br, jfabbio@ibest.com.br, katyusco@gmail.com

RESUMO

A automação sob todos os aspectos tem se difundido em escala elevada e promete representar o futuro para a Tecnologia da Informação, segundo alguns pesquisadores. Sistemas capazes de executar diversas ações são comandados por microcontroladores e microcomputadores. Nesse contexto, esse trabalho apresenta o estudo de caso sobre a construção de um sistema automatizado de controle de acesso a diversos ambientes, como requisito para a conclusão da disciplina de linguagem de programação orientada a objeto do curso de Tecnologia em Sistemas de Informação do CEFETPE. Reuniões e pesquisas convergiram para a utilização do microcontrolador PIC, da linguagem de programação Java e do banco de dados relacional como sendo alternativas viáveis à implementação. Inicialmente foi realizada uma especificação de requisitos para o sistema, em seguida sua implementação no conjunto de camadas: *hardware/firmware*, um microcontrolador com *software* embarcado cuja função é enviar um sinal de abertura para fechadura ou enviar um sinal indicando que não existe permissão; um *middleware* – aplicação Java utilizando as APIs JavaComm para se comunicar com o *hardware/firmware* e JDBC para se integrar a camada de base de dados do sistema. O sistema desenvolvido integrou artefatos de *software* com artefatos de *hardware,* apoiando-se em várias tecnologias a favor da automação.

Palavras-chave: automação, orientação a objetos, controle de acesso, desenvolvimento de sistema.

1. INTRODUÇÃO

A automação como um sistema automático pelo qual os mecanismos controlam seu próprio funcionamento, quase sem a interferência do homem está presente em áreas como indústrias, agricultura e demais áreas da atividade humana.

No núcleo de um sistema automatizado, de modo geral, encontra-se o microcontrolador que, por sua vez, pode-se definir como um "pequeno" componente eletrônico, dotado de uma "inteligência" programável, capaz de controlar periféricos, tais como: LED (*Light Emitting Diode*), botões, displays de segmentos, LCD (*Liquid Crystal Display*), relês, sensores e outros similares. Um microcontrolador é considerado um computador de pequeno porte, porque dentro deste chip encontra-se memória de programa, memória de dados, portas de entrada e saída, *timers*, contadores, conversores analógico-digitais, etc. (Souza, 2003).

Neste estudo de caso foi utilizado um microcontrolador interconectado a um componente de *software* através da API (*Application Programming Interfaces*) JavaComm, constituindo-se parte de um sistema de *software* com potencial para controlar o acesso de pessoas a diversos tipos de recintos: salas de aula, laboratórios, bibliotecas, entre outros. A outra parte do sistema diz respeito a integração com o banco de dados (Silberchatz et al., 1999) através da API JDBC (*Java Database Connectivity*), onde as regras de acesso podem ser gerenciadas através de uma interface gráfica construída com a API Swing, inicialmente, para inserir/apagar/modificar os dados das tabelas do modelo entidade relacionamento - MER.

2. MOTIVAÇÃO

Durante a disciplina de Linguagem de Programação Orientada a Objetos do curso de Tecnologia em Sistemas de Informação do CEFET-PE, um dos requisitos para sua conclusão, um projeto de *software* deveria ser construído desde que utilizasse os conceitos de Orientação a Objetos, linguagem de programação Java e suas mais variadas tecnologias. No perfil dos membros do grupo encontravam-se técnicos em Telecomunicações, Eletrônica e Eletrotécnica. Então, surgiu o desafio de se construir um projeto que pudesse aglutinar *hardware*, comunicação e *software*. O objetivo era não somente utilizar um microcontrolador como responsável de toda automação do processo, mas, também, sua interação com um *software* (aplicação para *desktop*), formando, assim, um sistema integrado com tratamento das informações através do processamento de dados originados a partir de uma entrada de dados oriunda de *hardware* que alimentasse um microprocessador e que este pudesse alimentar, também, um componente de *software*.

Depois de algumas sessões de *Brainstorm*, técnica para extração de requisitos utilizada largamente em Engenharia de Software (Pressman, 1995), o grupo chegou a conclusão que um Sistema de Controle de Acessos a recintos, ou seja, uma solução que restringisse o acesso de pessoas a determinado ambiente previamente autorizadas portadoras de uma senha individual, atendia aos requisitos desafiadores do projeto.

2.1. Arquitetura Física de Comunicação do Sistema

Tem-se, portanto, na motivação o desejo de realizar uma integração harmônica entre o *hardware* – módulo dispositivo controlador, e o *software* – aplicação no computador, deste modo, desenhou-se a arquitetura física de comunicação do sistema de controle de acesso (ver Figura 1).

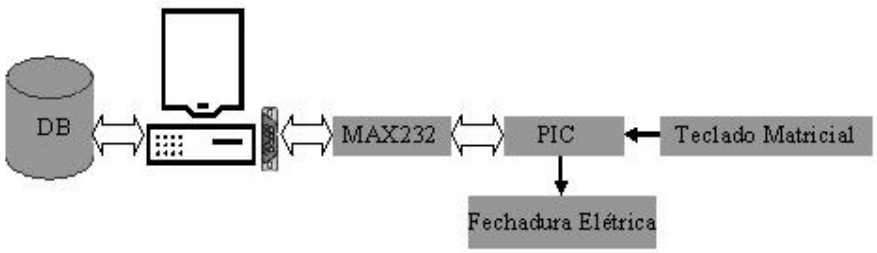

Figura 1 – Arquitetura Física de Comunicação

3. METODOLOGIA

Para a comunicação entre as duas partes, *hardware* e *software*, pesquisou-se a utilização da porta serial e o protocolo RS-232, considerando que se poderia obter alguma literatura específica, haja vista, ser um meio de comunicação bastante difundido entre projetos deste tipo. Para tanto, foi necessário se valer de algum

software auxiliar na comunicação com a porta serial e dentre os pesquisados escolheu-se a API do Java, Javacomm, pois esta fornece suporte às operações necessárias de comunicação e possibilidade de trabalhar com uma linguagem de programação orientada a objetos, deste modo, a aplicação Java foi desenvolvida utilizando esta API como interface para a porta serial.

Para as funções de supervisor do *hardware* desenvolvido foi utilizado o microcontrolador PIC, que apesar de possuir memória suficiente para armazenar uma fração de nomes de usuários de determinado ambiente ficou apenas responsável a enviar, controlar e receber dados de dispositivos e do computador.

E, de modo geral, o sistema e o hardware foram desenvolvidos dentro de uma concepção modular resultando numa arquitetura funcional como mostra a Figura 2.

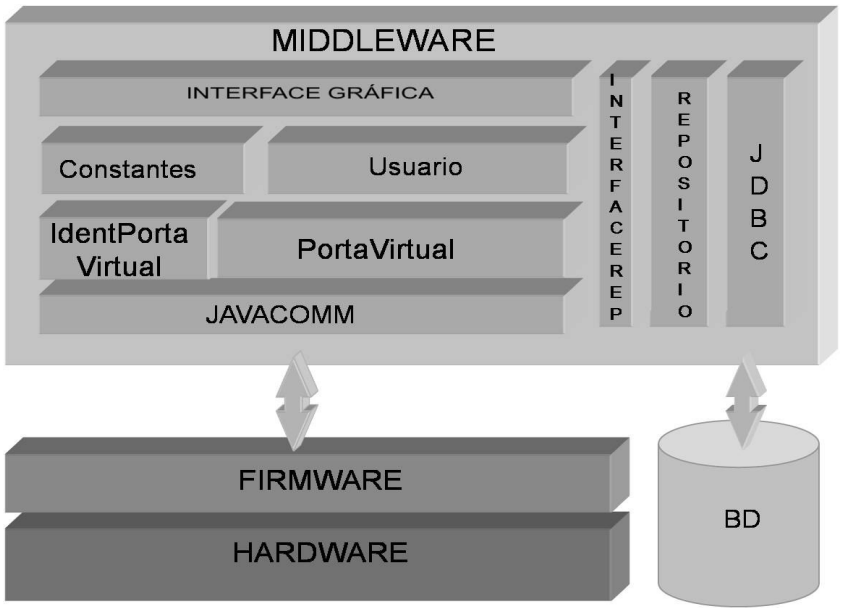

Figura 2 – Arquitetura Funcional

Abaixo estão descritos cada módulo da Figura 2:

- Hardware: composto de teclado matricial de 12 (doze) teclas e placa de circuito impresso com componentes como microcontrolador e o circuito integrado MAX232 (possibilita a comunicação via RS232 enviando uma seqüência de bits) conforme Figura 3. Faz interação com o usuário que acessa o ambiente controlado (através da digitação de sua senha) e é conectado ao computador através da porta de comunicação serial;

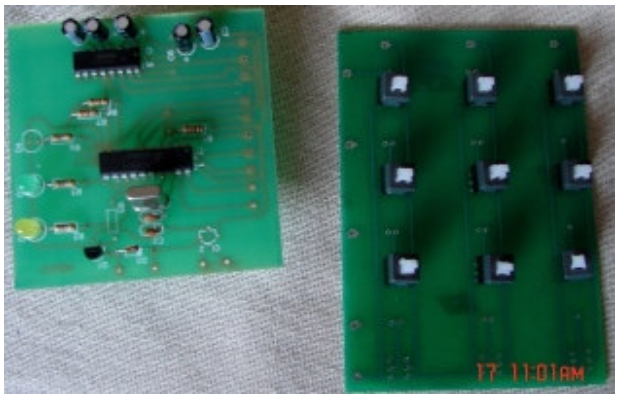

Figura 3 – Hardware Desenvolvido para o Controle de Acesso

- Firmware: programa embarcado no microcontrolador, cuja função é executar o controle efetivo do sistema do lado do *hardware*;

- Middleware: Composto de classes implementadas (Interface, Constantes, Usuario, IdentPortaVirtual, PortaVirtual, InterfaceRep e Repositorio) e de APIs (Javacomm e JDBC), esta camada é responsável pelo cadastro, consulta e manutenção dos usuários que acessam o ambiente controlado, suas classes, também, validam o acesso ao ambiente, recebendo as senhas digitadas no teclado matricial faz uma busca no banco de dados sobre determinado usuário, através da API JDBC que favorece o envio de instruções SQL para qualquer banco de dados relacional, e, este *middleware* baseado nas regras de validação permite a entrada ou não de pessoas sinalizando para o *hardware*, através da API Javacomm;
- Banco de dados: local onde ficam armazenados, de forma persistente, os dados dos usuários que acessam o ambiente.

4. FLUXOGRAMA DO CONTROLE DE ACESSO

O fluxograma da Figura 4 mostra de forma básica a ação principal do sistema de controle de acesso, Inicialmente, determinado usuário do ambiente tecla um conjunto de números no teclado matricial de 12 teclas, estes são enviados ao *middleware*. Então, existem duas possibilidades: 1 - Se for pressionado o conjunto de números mais a tecla * serão apagados os números enviados até então para o computador (os dados no *buffer* de leitura será limpo) e acenderá o LED vermelho e, o sistema ficará em estado de espera novamente; 2 - caso seja pressionado o conjunto de números mais a tecla # o sistema consultará se aqueles números fazem parte da base de dados: Se afirmativo – acenderá o LED verde (senha existe) e a porta será aberta, caso não esteja cadastrado será informado através do LED vermelho (senha inexistente).

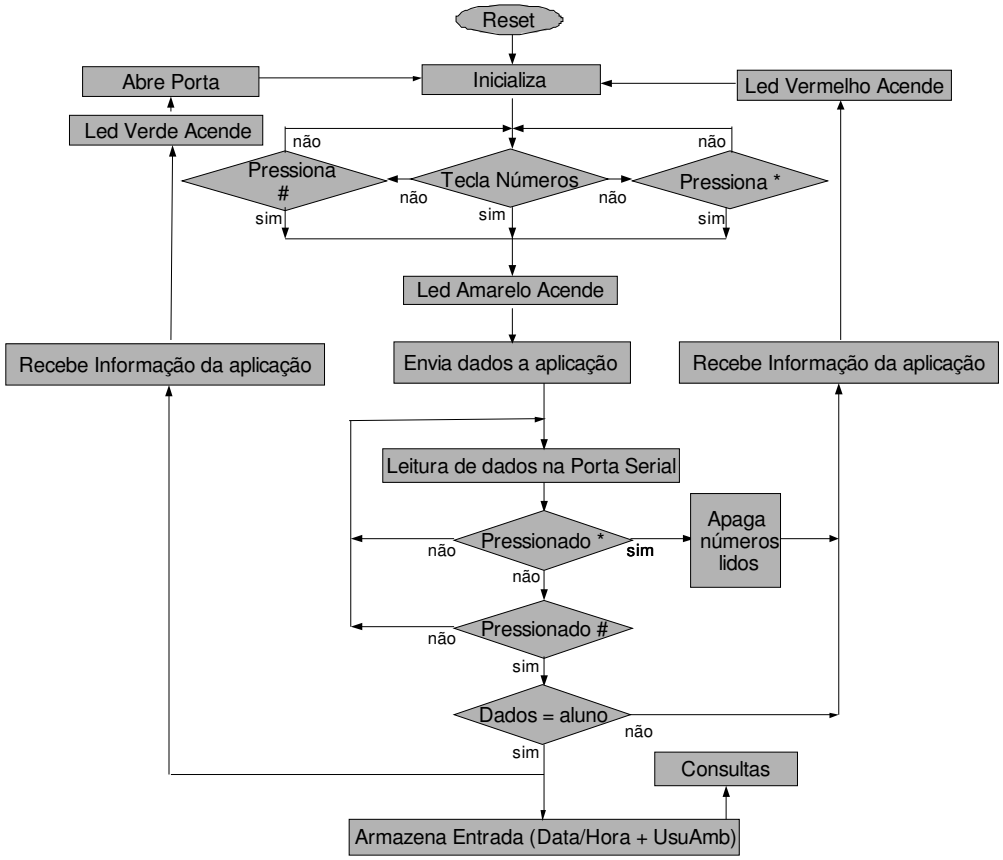

Figura 4 – Fluxograma Básico do Controle de Acesso

5. PROGRAMA EMBARCADO

O programa embarcado, também chamado de *firmware* – conjunto de regras que rege o funcionamento do sistema (Oliveira and Andrade, 2006), foi desenvolvido em linguagem de programação C, utilizando-se de um ambiente integrado para o estudo e desenvolvimento com a família PIC de microcontroladores – MPLAB.

O *firmware* é composto de um arquivo principal que se utiliza de nove bibliotecas, dentre as quais se destacam uma para controle da USART (módulo de comunicação serial do microcontrolador PIC), e outras duas para configuração das características do microncontrolador, sendo, a primeira predefinida como cabeçalho padrão pelo fabricante e a segunda programada para peculiaridades do projeto em questão.

Há abaixo trechos do código, do arquivo principal. Inicializa-se o microcontrolador com a função init_pic, que faz a configuração do PIC com as características desejáveis ao projeto. Depois, inicializam-se os PORT's A e B.

```
void main(void)
   {
            init_pic();
         PORTA = 0;
         PORTB = 0;
    ...
```

A partir daí, o programa fica em loop aguardando que algum usuário digite sua senha.

Na seqüência, temos a condicional if que vai "perceber" quando os pinos RA0 e RA3 do microcontrolador forem levados a nível zero, quando se pressiona a respectiva tecla. Tem-se um Delay de 100ms para evitar interferências por transientes de tensão existentes pelo pressionar da tecla. Acende-se um LED que indicará que a tecla está sendo pressionada e depois será enviado 49 em ASC (que equivale ao número decimal 1) para o computador através da serial. Na linha seguinte cria-se um loop para evitar que se enviem várias vezes o caractere, caso o usuário mantenha a tecla pressionada continuamente. Assim, continua o mesmo raciocínio para as demais teclas existente no teclado matricial:

```
while(1) {
          if (RA0 == 0 && RA3 == 0){     //Tecla 1
                  {
                 DelayMs(100);
                  RB7 = 1;
                  envia(49);
                  while(RA0 == 0);
                  RB7 = 0;
                  DelayMs(100);
                  }
```

Acima se mostra como os caracteres são enviados para que o computador possa validar a senha.

Agora, veremos o que acontece quando o computador envia seqüências (67 ou 69), para que o microcontrolador sinalize e abra a porta de acesso do ambiente controlado ou apenas sinalize senha incorreta, não permita o acesso.

Quando o microcontrolador recebe uma seqüência através da serial é feito com que ela seja igual a variável **recebido**. Caso esse valor seja 67, colocam-se os pinos RB6 e RB5 em nível alto, abrindo a fechadura e acendendo o LED que indica senha correta. Em seguida é temporizada esta condição anterior durante 150ms e por fim coloca-se RB6 e RB5 novamente em nível baixo e atribuí-se à variável **recebido** ao valor zero:

```
                  if (recebido == 67) {   //Corresponde a SENHA CORRETA.
                        RB6 = 1;
                        //Libera a fechadura.
                        RB5 = 1;
                        //Acende o LED Verde.
                        time = 0;
               // Zera o time. O time (tempo) vai estar sempre "Correndo".
                        while(time < 150);

               //Efetua a "ação" enquanto o tempo for menor que 150 ms.
                              RB6 = 0;
                        //Desliga a fechadura.
                              RB5 = 0;
                        //Apaga o LED Verde.
                              recebido = 0;
                        }
```

Por sua vez, caso recebido seja igual ao valor 69, coloca-se o pino RB4 em nível alto, acendendo um LED que indica senha incorreta. Em seguida é temporizada esta condição anterior durante 150ms e para finalizar coloca-se RB4 em nível baixo e atribuí-se à variável **recebido** ao valor zero:

```
if (recebido == 69){//Corresponde a SENHA INCORRETA.
      RB4 = 1;
      //Acende o LED Vermelho.
      time = 0;
      while (time < 100);
            RB4 = 0;
            //apaga o LED Vermelho
            recebido = 0;
      }
```

Em linhas gerais apresentamos acima trechos do código do programa embarcado que estão ligados às funções principais, executadas pelo sistema.

6. APLICAÇÃO DESENVOLVIDA

A aplicação desenvolvida no paradigma de orientação a objetos explorou a tecnologia Java voltada a *desktop J2SE – Java 2 Standard Edition* (Deitel and Deitel, 2005), esta aplicação desenvolveu-se no ambiente de programação *NetBeans*, pelo motivo desta ferramenta apresentar uma interface com vários componentes e propriedades de elementos como recursos de criação de formulários.

Os requisitos para implementação basearam-se em cenários possíveis a serem desenvolvidos dentro da concepção inicial, desta forma, fez-se necessário a comunicação com o meio externo ao computador, o sistema embarcado, e, sendo assim, utilizou-se como interface a porta de comunicação serial COM, através do protocolo RS232, isto foi possível valendo-se do auxílio da plataforma independente de comunicação do pacote Java, chamada Javacomm. A Figura 5 mostra os casos de uso elaborados neste projeto de controle de acesso, segundo Cardoso (2003) os casos de uso são conjuntos de funcionalidades de um sistema, representado por fluxos de eventos iniciados por um ator e apresentando um resultado de um valor a um ator. Na Figura 5 o ator dispositivo externo solicita acesso ao sistema de controle de acesso e o outro ator, usuário da aplicação, pode segundo um caso de uso consultar usuários ou, ainda, conforme outro caso de uso, este mesmo ator pode cadastrar usuários que acessam o ambiente.

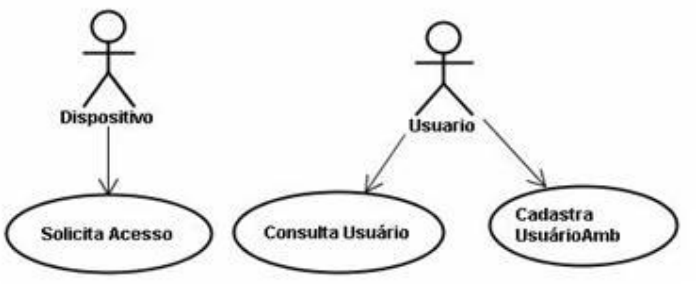

Figura 5 – Casos de Uso

6.1. Detalhes de Implementação

A aplicação esta dividida em pacotes e, por sua vez estes pacotes contém classes com funções específicas de acesso a dados, negocio e visão, as classes do pacote negocio são as que têm a inteligência – o poder de decisão deste sistema de controle de acesso. Entre todas as classes do pacote negociação a PortaVirtual é tratada como uma classe principal, pois esta envia sinais de controle ao meio externo e, também, recebe os dígitos do meio externo, esta classe possui três métodos de destaque.

Método LerPorta: Cria um *eventListner* e habilita notificação de dados disponíveis para permitir leitura de informações através da porta. Cria também uma *thread* cuja função é ficar *"escutando"* a porta continuamente, conforme trecho abaixo:

```
...
        try {
        porta.addEventListener(this);
        }catch (Exception e) {
        System.exit(1);
        }
        porta.notifyOnDataAvailable(true);
```

```
                try {
                threadLeitura = new Thread(this);
                threadLeitura.start();
        ...
```

Método Escrever: Recebe uma mensagem passada para o método e envia a mesma através da porta de comunicação:

```
public void escrever(String msg){
        if (escrita==true) {
                try {
                escrita = porta.getOutputStream();
                } catch (Exception e) {
                }
                try {
                        escrita.write(msg.getBytes());
                        escrita.flush();
                ...
```

Método EventoSerial: No início deste método existe um Switch onde pode-se indicar o que o método vai fazer em caso de diversos tipos de eventos seriais, sinal de controle, requisição para enviar e outros. Porém, na aplicação desenvolvida considerou-se apenas o evento que trata dos dados disponíveis – recebidos pela porta serial, pelo fato de está se usando apenas transmissão de dados da porta serial para o dispositivo microcontrolado e recepção de dados vindos do dispositivo microcontrolado para porta serial.

Criou-se um *buffer* para armazenar cada caractere que chega pela porta, então, analisa-se este caractere e é feito o seguinte: caso não seja igual a **enter** (tecla # do teclado pertencente ao *hardware* externo) ou a **cancela** (tecla * do teclado pertencente ao *hardware* externo), concatena-se o caractere com outros que já tenham chegado ou com outros que venham em seguida:

```
                ...
                byte[] bufferLeitura = new byte[1];
                try {
                        while ( leitura.available() > 0 ) {
                        nodeBytes = leitura.read(bufferLeitura);
                        }
                        String dadosLidos = new String(bufferLeitura);
                        if (senhaGeral != null)
                                senhaGeral = senhaGeral + dadoslidos;
                                else
                                senhaGeral = dadosLidos;

                        ...
```

Caso o dado recebido seja **cancela**, limpa-se o "buffer":

```
                        ...
                        if (dadosLidos.equals(Constantes.cancela)){
                                senhaGeral = "";
                                dadoslidos = "";
                        }
                        ...
```

Para **enter**, chama-se um método que verifica se os caracteres contidos no *buffer* fazem parte daqueles que constam no banco de dados, escreve **C** se conter o conjunto de caractere no banco e escreve **E** caso não contenha os caracteres no banco:

```
                        ...
                        if (dadosLidos.equals(Constantes.enter)){  //if 3
                                if (existeSenha1()){
                                        habilitarEscrita();
                                        identificaPorta();
                                        escrever("C");
                                        habilitarLeitura();
                                        senhaGeral = "";
                                        dadosLidos = "";
                                }
                        ...
```

C ou **E** são enviados através do método escrever, **C** equivale ao sinal para o microcontrolador abrir a porta, **E** equivale ao sinal para o microcontrolador acender o LED vermelho (indica senha digitada foi errada).

6.2. Interface Gráfica

Para uma melhor familiarização com o ambiente *Windows* foi criado com auxílio da API Swing formulários - interface gráfica do sistema, localizado no pacote visão da aplicação. Um dos formulários desenvolvidos o qual é composto de campos para entrada de dados de usuários do ambiente controlado é mostrado abaixo, Figura 6.

Figura 6 – Interface Gráfica para Cadastro

7. TRABALHOS FUTUROS

O sistema de controle de acesso possui um grande poder de expansão. Várias modificações podem ser implementadas de forma a atender uma série de requisitos que venham a ser solicitados. Relatamos algumas alterações que podem ser adicionadas a partir do modelo inicial aqui colocado, porém, outras possibilidades ainda não vistas podem ser analisadas:

- Criação de um *log* de acesso, onde se podem visualizar as pessoas que freqüentaram o ambiente controlado em determinadas datas;
- Acréscimo no conjunto dos números de matrícula no cadastro do usuário, permitindo que possa aumentar a quantidade de informações necessárias ao acesso do ambiente (Ex.: Alunos do terceiro módulo de determinado curso só acessam o laboratório no horário da noite – O número da matrícula pode permitir saber se a pessoa é aluno do módulo especificado);
- Criação de outra camada na aplicação desktop, possibilitando a interligação com a internet. Desta forma o administrador poderá, por exemplo, acessar a consultas e relatórios de acesso e saber quem entrou na sala em um dia qualquer através da Web (rede de computadores de alcance mundial);
- Alterações que permitam a instalação de sensores de presença ao sistema de forma que, quando digitado um determinado código de ativação destes sensores de monitoramento se houver alguma interceptação um alarme será acionado;
- Modificações que permitam o uso de comunicação com o meio externo através da utilização da porta USB (Universal Serial Bus);
- Outros recursos no banco de dados como *procedures* e gatilhos para cadastro de usuários e consultas, melhorias nas regras de validação, a permitir que na consulta ao banco sejam considerados alguns filtros, por exemplo, dias ou horas que podem ser liberados para entrada de usuários a determinados ambientes.

Como se vê, a utilização da orientação a objetos, banco de dados e de microcontroladores, de forma modular, permite que adequações ou ampliações que venham a ser necessárias possam sim se tornar realidades.

8. CONSIDERAÇÕES FINAS

Com o uso da linguagem de programação Java foi possível visualizar a solução em níveis e se valer de várias ferramentas como APIs de acesso a banco de dados, de acesso a porta serial e outros pacotes próprios que o paradigma de orientação a objetos nos oferece. Desta forma, acredita-se que outros padrões de programação, como a linguagem procedural, não trariam os benefícios que a linguagem Java nos ofereceu, no entanto, houve um esforço inicial na utilização da linguagem C e C++ para desenvolvimento da aplicação desktop, porém, foram descartadas diante da gama de recursos proporcionado pelo Java.

Um ponto importante na aplicação Java é que existe uma interface gráfica para cadastro e consultas, e ainda concomitantemente esta mesma aplicação, de forma independente, realiza sua função principal de controle de acesso, lendo dados da porta serial e validando-os, isto, em todo o tempo de ativação do sistema.

Outro ponto importante do ponto de vista do projeto foi a adesão ao microcontrolador PIC, por este ser versátil e financeiramente mais viável em relação a outros microcontroladores como os da família 8051.

Finalmente aponta-se para conquista de um objetivo, proposto e realizado com a junção de várias tecnologias, principal objetivo, tendo em vista o desejo do conhecimento interdisciplinar.

REFERÊNCIAS

API Javacomm. <http://java.sun.com/products/javacomm/reference/api/index.html> Acesso em: 20 mar 2007.

BOYLESTAD, R. L.; NASHELSKY, L. **Dispositivos Eletrônicos e Teoria de Circuitos**. 6. ed. Rio de Janeiro: Prentice-Hall do Brasil, 1998.

Cardoso, C. **UML na Prática: do Problema ao Sistema**. Rio de Janeiro: Ciência Moderna, 2003.

Deitel, H. M.; Deitel, P. J. **Java como Programar.** 6. ed. São Paulo: Pearson Prentice Hall. 2005.

MICROCHIP. **Microcontrollers with nanoWatt Technology: PIC16F627A/628A/648A Data Sheet**. USA.

MPLAB. <http://www.ic.unicamp.br/~pannain/mc404/aulas/microcontroladores/pdf/MPLAB.PDF> Acesso em: 22 abr 2007.

NICOLOSI, D. E. C.; RODRIGO, B. B. **Microcontrolador 8051 com Linguagem C: Prático e Didático – Família AT89S8252 ATMEL**. São Paulo: Érica, 2005.

OLIVEIRA, A. S.; ANDRADE, F. S. **Sistemas Embarcados Hardware e Firmware na Prática.** São Paulo: Érica, 2006.

PATTERSON, D. A.; HENNESSY, J. L. **Organização e Projeto de Computadores – A Interface Hardware/Software**. 2. ed. Rio de Janeiro: LTC, 2000.

PRESSMAN, R. S. **Engenharia de Software**. São Paulo: Makron Books, 1995.

SILBERCHATZ, A.; KORTH, H. F.; SUNDERSHAN, S. **Sistema de Banco de Dados**. São Paulo: Makron Books, 1999.

SILVEIRA, J. L. **Comunicação de Dados e Sistemas de Teleprocessamento.** São Paulo: Makron Books, 1991.

SOUZA, D. J. **Desbravando o PIC: ampliado e atualizado para 16F628A**. São Paulo: Érica, 2003.

TEXAS INSTRUMENT. **MAX232, MAX232I Dual EIA-232 Drivers/Receives: Data Sheet**. USA.
<WEB, http://pt.wikipedia.org/wiki/WEB> Acesso em: 22 abr 2007.