MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL FLUMINENSE
CENTRO TECNOLÓGICO
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA DE TELECOMUNICAÇÕES

CONTROLE DE ACESSO

Bruno Leite Diniz

Orientador: Luiz Cláudio Schara Magalhães

Niterói
2º semestre / 2005

Aos meus pais, que me ajudaram nesta longa caminhada, e a todos aqueles que contribuíram para a minha formação.

RESUMO

Foi desenvolvido um software que permite, através de um computador, a abertura de uma fechadura eletrônica possibilitando, assim, o controle de acesso a um determinado ambiente. O software é baseado no banco de dados dos usuários de um sistema Linux, que por sua vez, controla o acesso mediante um login e senha.

Palavras-chave: controle de acesso, login do Linux, fechadura eletrônica, usuários do Linux, comunicação serial, Iopener.

ÍNDICE

ÍNDICE DE ILUSTRAÇÕES

1. INTRODUÇÃO

Disponibilizar o acesso a um determinado local, pode vir a gerenciar desde uma porta até um sistema de controles envolvendo catracas, cancelas e portas. A inconveniência de carregar chaves, dispositivos eletrônicos de abertura ou cartões utilizados nos leitores de código tomou a atenção para a criação de um controle de acesso livre de tais objetos.

Sistemas de controle de acesso sem a necessidade de carregar dispositivos ou objetos já são largamente utilizados no mercado, por tal motivo, fazer mais um, não seria novidade. A idéia foi criar um sistema de:

- baixo custo, tanto pelo fato de não usar chaves ou cartões, seja quantos forem o número de usuários, quanto pelo criação do sistema;

- fácil implantação, já que alguns sistemas apresentam manuais exaustivos de instalação e cadastro dos usuários;

- fácil gerenciamento, pelo mesmo motivo anterior, só que no controle do acesso dos usuários.

2. SOLUÇÃO

Baseando somente na criação de um software e um banco de dados, pode-se ter tal sistema, já que o custo de software é relativamente baixo, tanto para a confecção do programa, quanto para a aquisição do usuário final. Porém, deixaria de ser de fácil implantação, uma vez que teríamos que cadastrar todos aqueles, que utilizarão o sistema, neste novo banco de dados.

A intenção de fazer o sistema de controle de acesso por programação faz restar apenas uma característica a ser cumprida, o que mostra o caminho a ser seguido. A nova idéia foi, então, de associar ao controle de acesso ao banco de dados de usuários do sistema operacional Linux.

Este sistema é amplamente utilizado em universidades, colégios, empresas e residências, e provavelmente, aqueles, que freqüentam esses ambientes e suas salas, já tem algum tipo de acesso no sistema, como login e senha.

3. MICROCONTROLADOR

A solução, inicialmente, apresentada foi, a utilização de um microcontrolador para interfacear um teclado de computador com a porta serial do próprio computador. Desta forma, não haveria a necessidade do computador ficar próximo do dispositivo, uma vez que a comunicação serial suporta maiores distâncias, o servidor de controle de acesso poderia até, então, permanecer numa outra sala. As funcionalidades do microcontrolador permitiriam, também, o acionamento da fechadura eletrônica e apresentação do estado do processo de controle, como leds, para indicação de login correto ou incorreto e senha correta ou incorreta, por exemplo.

Porém, esta solução foi descartada devido à dificuldade de ter que construir um mapa de teclado no microcontrolador, a fim de capturar as entradas digitadas e enviar para o computador terminar o processo de controle de acesso.

4. IOPENER

O Iopener foi um computador lançado nos EUA, com um sistema próprio, a baixo custo subsidiado, pois o acesso à internet por modem discado era, necessariamente, feito através do provedor da empresa, ganhando, assim, com serviços e alguns outros agregados.

Aconteceu que as pessoas ao adquirir este computador, utilizavam formas de trocar o sistema operacional vindo de fábrica e usavam este como computador pessoal comum, sem a obrigatoriedade mencionada acima. A empresa faliu e estes computadores são encontrados a venda na internet a preços bem baixos, pois já se passaram alguns anos e evolução tecnológica prosseguiu.

Por este motivo, foi adquirido e utilizado o Iopener, como próxima solução, para ser a interface de entrada dos dados, login e senha, no projeto de controle de acesso.

Figura 1 – Iopener

4.1 Iopener: instalação do sistema Linux

Primeiramente, foi necessário adquirir um Linux que servisse ao Iopener. Desde o lançamento deste computador, foram surgindo algumas versões que, apesar de seu hard disk ser uma memória flash de apenas 16MB, suprem os recursos disponíveis deste computador.

Foram encontrados na internet o LTSP (linux terminal server project), o Midori e o Jailbait, com imagens e programas fonte. Optou-se pelo Jailbait para o Iopener, produzido pelo Sourceforge.net. Segue, abaixo, o processo de instalação da imagem do Jailbait para a flash do Iopener:

- retire a parte traseira do Iopener;

Figura 2 – parte traseira do Iopener

- localize a interface para conexão do cabo ide de laptop;

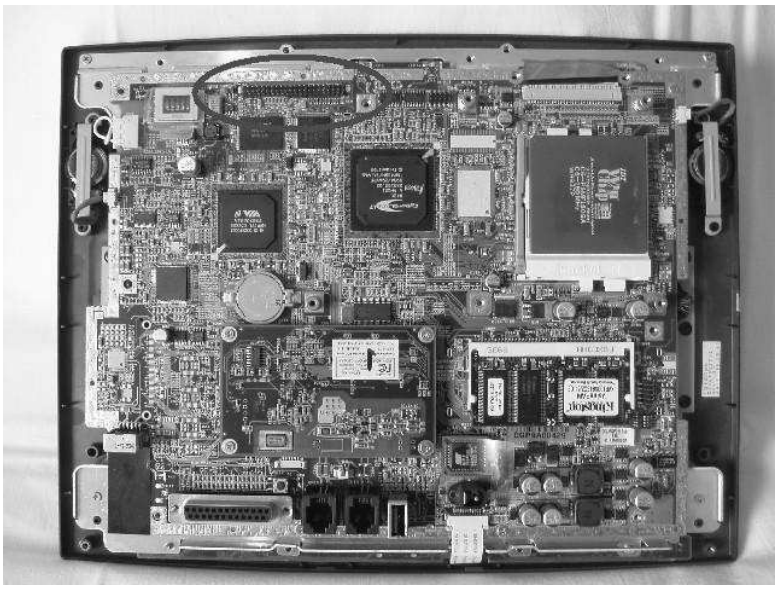

Figura 3 – conexão ide para laptop

- encaixe o cabo ide, com um hard disk de laptop na outra ponta, previamente instalado um sistema Linux;

Figura 4 – hard disk externo de laptop

- neste hard disk de laptop, também já deve conter a imagem do Jailbait;
- ligue o Iopener e acesse pelo hard disk externo, simplesmente dando boot pelo drive c;
- descompacte a imagem:

gunzip version6_fullinstall.img.gz

- instale a imagem na flash do Iopener:

dd if=version6_fullinstall.img of=/dev/hdb

- desligue a máquina;
- desconecte o cabo ide do Iopener;
- ligue a máquina.

A empresa que produzia o Iopener, como forma de impossibilitar a transformação para um computador livre de seu sistema, começou a se valer de alguns artifícios. Na hora de colocar um hard disk externo, deve-se selecionar este como boot principal. O problema que uma das coisas que a empresa fez, foi fazer com que, ao colocar este hard disk externo, travasse a bios do computador. Como foi o caso, antes de todo o procedimento acima, foi colocado uma bios liberada no lugar da original. Dessa forma foi possível o processo descrito.

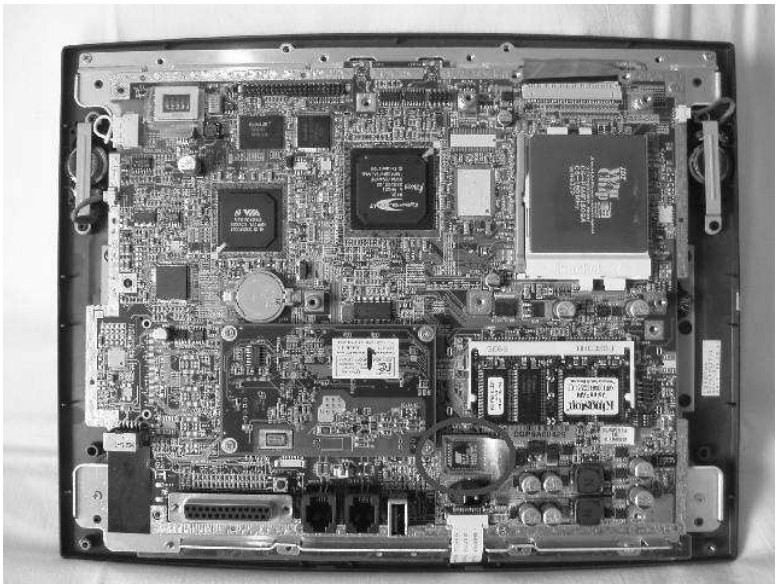

Figura 5 – bios trocada

5. CLIENTE / SERVIDOR

Foram feitos, dois programas, um cliente e um servidor, que usam a porta serial para a comunicação e fazem o controle de acesso.

O programa cliente, que é utilizado como interface com o usuário e que é instalado no Iopener, somente recebe o login e senha deste e envia, pela porta serial, para o servidor.

O programa servidor é responsável por processar os dados recebidos e apresentar uma resposta: usuário existe, usuário não existe, senha correta, senha incorreta, etc. Neste processamento, é que são utilizadas as bases de dados dos usuários cadastrados naquele sistema Linux, facilitando o gerenciamento do controle de acesso, uma vez que a utilização deste sistema é comum à maioria dos administradores de rede.

5.1 Cliente / servidor: software

São apresentados, abaixo, os programas cliente e servidor. O programa cliente foi instalado no iopener, que ficaria do lado de fora do ambiente que se deseja o acesso, e o programa servidor no computador com o sistema linux, dentro do ambiente, onde estão cadastrados os usuários do sistema.

5.1.1 Programa cliente

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <shadow.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <errno.h>
static int fd=0;
char buffer[64], *serial;
struct termios tio_antigo,tio_novo;

int fechar_porta_com(){
        printf(" Fechando porta serial...\n\n");
        int fechar;
        fechar=close(fd);
        return(fechar);
}

int abrir_porta_com(){
        printf(" Abrindo porta serial...\n");
        fd=open("/dev/ttyS0", O_RDWR | O_NOCTTY);
        if(isatty(fd)) serial=ttyname(fd);
        tcgetattr(fd,&tio_antigo);
        bzero(&tio_novo,sizeof(tio_novo));
        tio_novo.c_cflag = CLOCAL | CREAD | CS7 | B38400 | PARENB;
        tio_novo.c_oflag = 0;
        tio_novo.c_iflag = IXON | IXOFF;
        tio_novo.c_lflag = 0;
        tio_novo.c_cc[VTIME] = 10;
        tio_novo.c_cc[VMIN] = 64;
        tcflush(fd,TCIFLUSH);
        tcsetattr(fd,TCSANOW,&tio_novo);
        return(fd);
}

int escrever_porta_com(){
        printf(" Escrevendo na porta serial...\n");
        ssize_t saida;
        saida=write(fd,(void *) &buffer,sizeof(buffer));
        printf(" Escrito %d caracteres: %s\n",saida,buffer);
        return(saida);
}
```

```
int ler_porta_com(){
        printf(" Lendo porta serial...\n");
        ssize_t entrada;
        entrada=read(fd,(void *) &buffer,sizeof(buffer));
        printf(" Lido %d caracteres: %s\n",entrada,buffer);
        return(entrada);
}

int main(int argc,char *argv[]){
        printf("\n * * * * CPPC * * * *\n");
        if(abrir_porta_com()==-1) printf(" Erro em abrir(): %d %s\n",errno,strerror(errno));
        bzero((void *) &buffer,sizeof(buffer));
        if(argc<2) buffer[64]='\0';
        else strcpy(buffer,argv[1]);
        if(escrever_porta_com()==-1) printf(" Erro em escrever(): %d %s\n",errno,strerror(errno));
        sleep(1);
        bzero((void *) &buffer,sizeof(buffer));
        if(argc<3) buffer[64]='\0';
        else strcpy(buffer,argv[2]);
        if(escrever_porta_com()==-1) printf(" Erro em escrever(): %d %s\n",errno,strerror(errno));
        if(ler_porta_com()==-1) printf(" Erro em ler(): %d %s\n",errno,strerror(errno));
        tcsetattr(fd,TCSANOW,&tio_antigo);
        if(fechar_porta_com()==-1) printf(" Erro em fechar(): %d %s\n",errno,strerror(errno));
        return(0);
}
```

Seguindo a lógica do main(), temos:

- na função abrir_portacom(), é escolhido e inicializado a porta serial de comunicação;
- limpa-se o buffer, para não haver erro da cópia do login;
- então, envia-se o nome através da função escrever_porta_com();
- espera-se um segundo até o ler_porta_com(), que é a função que recebe os dados, se recomponha no lado do servidor;
- novamente, limpa-se o buffer para não haver erro, agora, da cópia da senha;
- envia-se a senha através da serial pela função escrever_porta_com();
- o programa fica na função ler_porta_com(), esperando a resposta pelo lado do servidor;
- encerra-se a comunicação serial do programa cliente.

5.1.2 Programa servidor

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <shadow.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <errno.h>
#include <crypt.h>
static int fd=0;
char buffer[64], *serial,usuario[64],senha[64];
struct termios tio_antigo,tio_novo;

int fechar_porta_com(){
	printf(" Fechando porta serial...\n\n");
	int fechar;
	fechar=close(fd);
	return(fechar);
}

int abrir_porta_com(){
	printf(" Abrindo porta serial...\n");
	fd=open("/dev/ttyS0", O_RDWR | O_NOCTTY);
	if(isatty(fd)) serial=ttyname(fd);
	tcgetattr(fd,&tio_antigo);
	bzero(&tio_novo,sizeof(tio_novo));
	tio_novo.c_cflag = CLOCAL | CREAD | CS7 | B38400 | PARENB;
	tio_novo.c_oflag = 0;
	tio_novo.c_iflag = IXON | IXOFF;
	tio_novo.c_lflag = 0;
	tio_novo.c_cc[VTIME] = 10;
	tio_novo.c_cc[VMIN] = 64;
	tcflush(fd,TCIFLUSH);
	tcsetattr(fd,TCSANOW,&tio_novo);
	return(fd);
}

int escrever_porta_com(){
	printf(" Escrevendo na porta serial...\n");
	ssize_t saida;
	saida=write(fd,(void *) &buffer,sizeof(buffer));
	printf(" Escrito %d caracteres: %s\n",saida,buffer);
	return(saida);
}
```

```
int ler_porta_com(){
	printf(" Lendo porta serial...\n");
	ssize_t entrada;
	entrada=read(fd,(void *) &buffer,sizeof(buffer));
	printf(" Lido %d caracteres: %s\n",entrada,buffer);
	return(entrada);
}

void verificar(){
	printf(" Verificando usuário e senha...\n");
	char salt[11],crypt_password[34];
	struct spwd *spwd=NULL;
	salt[11]='\0';
	crypt_password[34]='\0';
	spwd=getspnam(usuario);
	if((spwd!=NULL)&&(spwd->sp_pwdp!='\0')&&(senha!='\0')){
		strncpy(salt,spwd->sp_pwdp,sizeof(salt));
		strcpy(crypt_password,crypt(senha,salt));
		if(strcmp(crypt_password,spwd->sp_pwdp)==0) strcpy(buffer,"Senha correta!");
		else strcpy(buffer,"Senha incorreta!");
	}
	else if(senha=='\0') strcpy(buffer,"Falta senha!");
	else strcpy(buffer,"Usuário inexistente!");
}

void abrir_tranca(){
	ioctl(fd,TIOCMSET,TIOCM_CTS);
}

int main(){
	printf("\n * * * * CPPC * * * *\n");
	bzero((void *) &buffer,sizeof(buffer));
	bzero((void *) &usuario,sizeof(usuario));
	bzero((void *) &senha,sizeof(senha));
	if(abrir_porta_com()==-1) printf(" Erro em abrir(): %d %s\n",errno,strerror(errno));
	if(ler_porta_com()==-1) printf(" Erro em ler(): %d %s\n",errno,strerror(errno));
	strcpy(usuario,buffer);
	bzero((void *) &buffer,sizeof(buffer));
	if(ler_porta_com()==-1) printf(" Erro em ler(): %d %s\n",errno,strerror(errno));
	strcpy(senha,buffer);
	verificar();
	sleep(1);
	if(escrever_porta_com()==-1) printf(" Erro em escrever(): %d %s\n",errno,strerror(errno));
	abrir_tranca();
	tcsetattr(fd,TCSANOW,&tio_antigo);
	if(fechar_porta_com()==-1) printf(" Erro em fechar(): %d %s\n",errno,strerror(errno));
	return(0);
}
```

Também, seguindo a lógica do main(), temos:

- limpam-se todos os buffers;
- na função abrir_portacom(), é escolhido e inicializado a porta serial de comunicação;
- entra na função ler_porta_com() e fica aguardando uma chegada de dados;
- copia-se do buffer para liberá-lo para a chegada da senha;
- entra na função ler_porta_com() e fica aguardando uma chegada de senha;
- verifica-se o estado do usuário solicitado na função verificar(), ou seja, se o login existe e se a senha está correta;
- envia-se uma resposta para o programa cliente, através da função escrever_porta_com(), de forma que o usuário fique sabendo o estado;
- caso a função verificar() chegue ao ponto de apresentar no buffer a string “Senha correta!”, então, a função abrir_tranca() abre a porta para o acesso;
- encerra-se a comunicação serial do programa servidor.

6. CONTROLE DE ACESSO

Com a preocupação do Iopener permanecer do lado externo do ambiente e com as possibilidade que um Linux completo poderia apresentar como alternativas de futura evolução do programa, optou-se por colocar um computador comum como interface de entrada de dados para o controle de acesso, onde somente o teclado ficaria do lado de fora.

Com a solução descrita, não há mais a necessidade de uma comunicação serial entre dois programas, então, esta será a interface do computador utilizada para criar uma diferença de potencial em um de seus pinos, de forma a liberar o acesso dos usuários.

Essa tensão é ocasionada apenas executando o programa final apresentado no próximo item, uma vez que o Linux foi modificado no seu arquivo /etc/passwd para executar este programa de controle de acesso e não o sistema.

Dessa forma, também, foi descartada a função verificar() descrita no capítulo anterior e, assim, faz-se uso do próprio login do Linux com suas funcionalidades de acesso, que podem, então, ser aproveitadas, como não conseguir acesso por login expirado ou rotular o usuário a determinado grupo.

6.1 Programa final comentado

Segue o programa utilizado para o controle de acesso dos usuários:

```
/*
 * Universidade Federal Fluminense
 * Engenharia de Telecomunicações
 * Trabalho Final do 2º semestre de 2005
 * Título: Controle de Acesso
 * Objetivo: Controlar o acesso a um ambiente através do login do Linux
 * Aluno: Bruno Leite Diniz
 * Orientador: Luiz Cláudio Schara Magalhães
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/time.h>
#include <time.h>

static int fd=0;
char buffer[64],*usuario;
struct timeval tv;
time_t hora;
FILE *stream;

void abrir_porta_com(){
        printf("\n Acionando tranca...\n");
        fd=open("/dev/ttyS0",O_RDWR | O_NOCTTY); //abre conexão
        ioctl(fd,TIOCMSET,TIOCM_CTS); //levanta de -12v para +12V o pino DSR(6) e CTS(8)
}

void fechar_porta_com(){
        printf("\n Desligando acionamento...\n\n");
        close(fd); //fecha conexão
}

void escrever_log(){
        stream=fopen("/var/log/ca_log","a"); //cria arquivo ca_log em /var/log
        hora=time(NULL); //obtem a hora
        bzero((void *) &buffer,sizeof(buffer)); //limpa buffer
```

```
        sprintf(buffer,"%s ----------> %s",getlogin(),ctime(&hora)); //escreve o login do usuário e a
hora formatada para o buffer
        fwrite(buffer,1,strlen(buffer),stream); //escreve do buffer para o arquivo
}

void abrir_tranca(){
        abrir_porta_com(); //função para abrir a comunição serial
        tv.tv_sec=0;
        tv.tv_usec=400000; //seleciona 0,4 segundo para a função select
        select(0,NULL,NULL,NULL,&tv); //temporizador para manter a tensão
        fechar_porta_com(); //função fechar a comunicação serial
}

int main(){
        printf("\n * * * * Controle de Acesso * * * *\n");
        abrir_tranca(); //função para abrir a tranca eletrônica
        escrever_log(); //função para criar um arquivo de log
        return(0);
}
```

Na função abrir_tranca(), estabelece-se uma comunicação serial a fim de poder usar a função ioctl(), gerando um sinal de +12V no pino 6 (DSR) para acionar o dispositivo eletrônico, que será apresentado no próximo capítulo. Em seguida, apresenta-se a função escrever_log(). De modo a ter um controle de entrada dos usuários, é criado um arquivo ca_log no diretório /var/log e inserido o login e a hora que o usuário liberou o acesso ao ambiente.

Alguns procedimentos foram executados na implantação do programa, que, como registro, seguem abaixo:

- executar a permissão para uso da porta serial para os demais usuários:

  chmod a+rw /dev/ttyS0

- trocar de /bin/bash para /ca no arquivo /etc/passwd para o programa a ser carregado na inicialização, ao invés do sistema;
- também executar a permissão para uso do arquivo de log, o /var/log/ca_log mencionado acima, para os demais usuários:

  chmod a+rw /var/log/ca_log

7. INTERFACE COMPUTADOR / FECHADURA

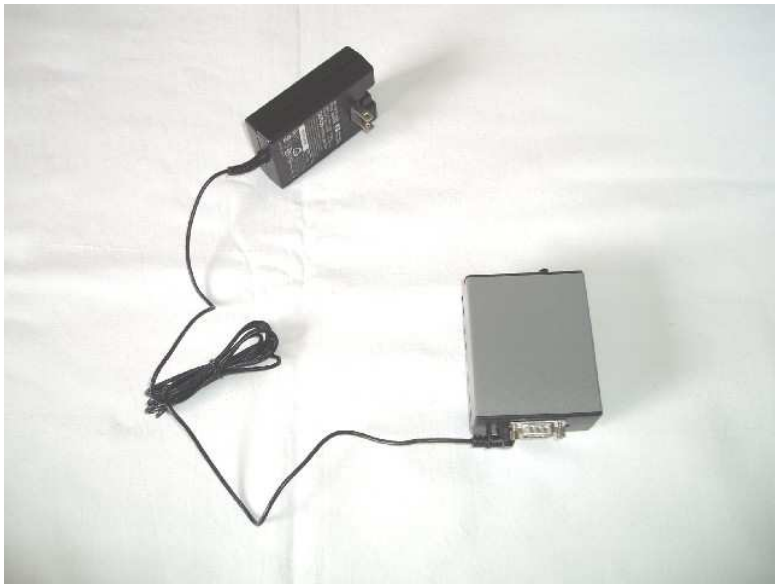

Figura 6 – dispositivo de interfaceamento computador / fechadura

Através de função abrir_tranca() é acionado a porta serial que tem como principio de comunicação manter uma tensão de -8V a -14V para o lógico 1 e +8V a +14V para o lógico 0 no fio, para a maioria das aplicações do padrão RS232. Dessa forma, esta função liga e desliga esta comunicação servindo de interuptor para a abertura da porta, no caso do procedimento de acesso do usuário descrito no item anterior for bem sucedido. A tensão no cabo serial aciona um relé que libera o diferencial de 12V da fonte da tranca automática. O sinal da porta serial não aciona diretamente a tranca a fim de proteger a interface do computador.

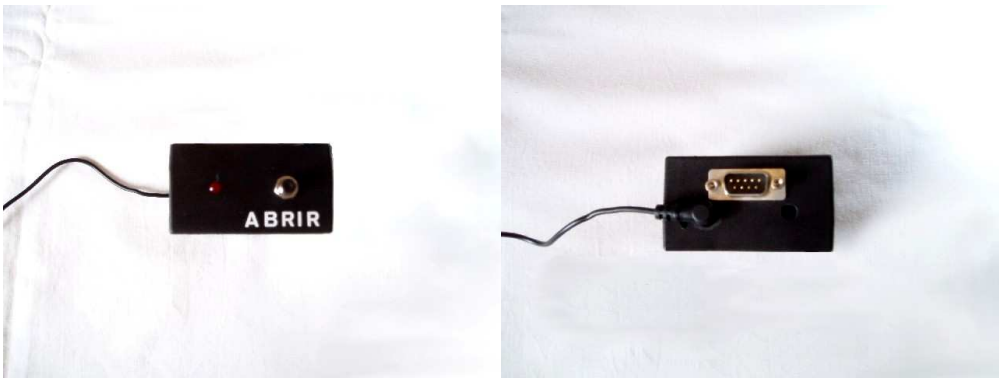

Figura 7- vista superior e inferior

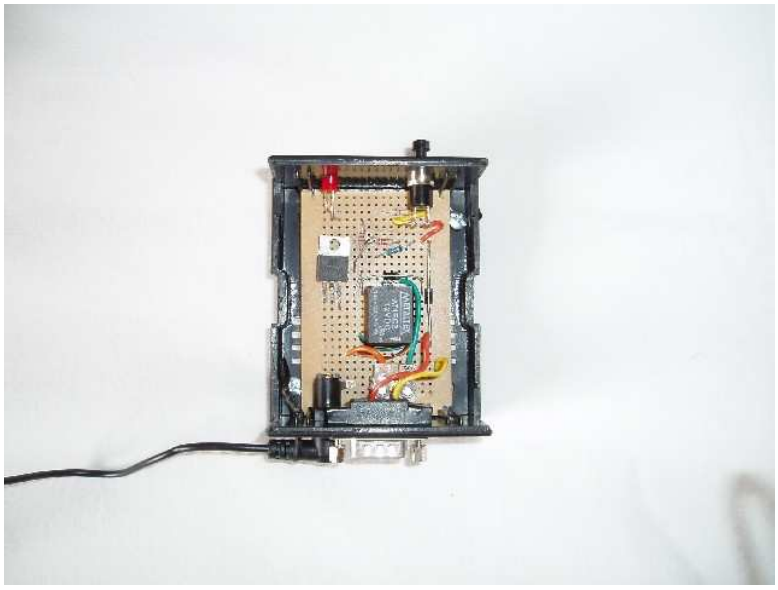

Figura 8 – circuito eletrônico de acionamento da tranca

Segue, abaixo, o esquema eletrônico da interface computador/fechadura para o acionamento da tranca:

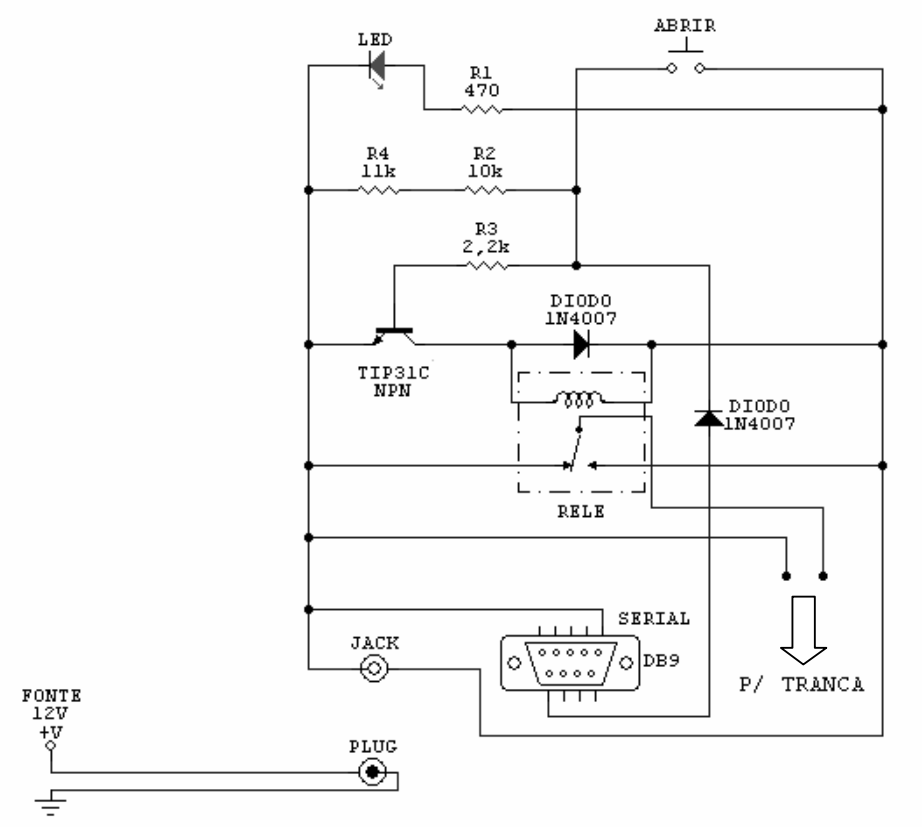

Figura 9 – esquema de acionamento computador/fechadura

Características do circuito:

- o led é utilizado somente para indicar que o dispositivo está ligado;

- os resistores de 11KΩ e 10KΩ tem a função de não deixar o circuito flutuando, a fim de eliminar quaisquer sinais espúrios provenientes do cabo serial, por isso colocados em paralelo com a base do transistor ao terra;

- tais valores foram usados por conveniência, porém valores de ordem superior ao resistor de 2,2KΩ, utilizado na polarização da base, serve para impedir a flutuação do circuito;

- o diodo em série com o pino 6 da serial serve proteger a interface do computador de qualquer sinal que pudesse vir a ser retornado pelo cabo;

- o pino 5 é ligado ao terra, segundo o padrão RS232;

- o transistor, quando polarizado pelo sinal serial ou pelo acionamento do botão abrir, abre a tranca;

- somente o transistor seria suficiente para a ddp na fechadura, porém como a maioria encontrada foi de baixa potência, utilizou-se um relé para o acionamento, caso contrário, despreze tal componente;

8. CONCLUSÃO

Como o sistema de controle de acesso foi feito na linguagem de programação C e no sistema operacional Linux, é surpreendente as possibilidades e alternativas para o acesso dos usuários. Horário de acesso; bloqueio do acesso por login expirado, ou seja, o usuário só tem o acesso para aquele semestre, por exemplo; acionamento de uma câmera na hora da entrada; criação de arquivos de log com data e horário do acesso; e etc, são algumas destas possibilidades. Outras funcionalidades podem vir a surgir com as necessidades do ambiente, o que, através do proposto pelo projeto, não precisa ser necessariamente de alto custo ou de gerenciamento dispendioso.

9. BIBLIOGRAFIA

- Maurício Cardoso de Sá, Programação C para Microcontroladores 8051, Editora Érica, 2005;

- Denys E. C. Nicolosi e Rodrigo B. Bronzeri, Microcontrolador 8051 com Linguagem C – Família AT89S8252 Atmel, Editora Érica, 2005;

- Fábio Pereira, Microcontroladores PIC – Programação em C, Editora Érica, 2005;

- Leonardo Marcilio Schunk e Aldo Luppi, Microcontroladores AVR – Teoria e Aplicações Práticas, Editora Érica, 2001;

- David José de Souza, Desbravando o PIC – Ampliado e Atualizado para PIC16F628A, Editora Érica, 2005;

- http://jailbail.sourceforge.net

- http://www.ltsp.org

- http://www.gnu.org

- http://focalinux.cipsga.org.com.br

- http://www.badflash.com