Relatório Final - Sistemas Operacionais Embarcados

Guarda-volumes eletrônico

Gabriel B. Pinheiro
Engenharia Eletrônica
Universidade de Brasília - Faculdade do Gama
Brasília-DF, Brasil
gabrielbopi@gmail.com

Elisa Costa Lima
Engenharia Eletrônica
Universidade de Brasília - Faculdade do Gama
Brasília-DF, Brasil
eliss.liima@gmail.com

Abstract*—Este documento descreve o desenvolvimento do projeto final da Disciplina de Sistemas Operacionais Embarcados.** ***Palavras-Chave—guarda-volumes; praticidade; segurança; Raspberry Pi; Servidor.

I Resumo

Para o presente documento, foi proposto do desenvolvimento de um guarda-volumes eletrônico. Este sistema se trata de um compartimento com tranca que é controlada de acordo com contas cadastradas num banco de dados num servidor central que faz a autenticação desses dados para o acesso seguro.

Esse sistema seria autônomo com teclado e tela embutidos para acesso do usuario, num terminal. Ao se ocupar o espaço interno reservado do guarda-volumes para o inventário de de um usuário, o terminal é interditado, operando num estado que espera a volta do usuário para autenticar e abrir novamente a tranca, e pegar de volta seu pertence.

Também foi preterido o aumento da camada de segurança através da leitura da impressão digital do usuário,também, ao se fazer o acesso em terminal.

Também para aumento da praticidade de uso, era almejada a emissão de um QRcode para o usuário, onde ele poderia exibi-lo à câmera do terminal para a automatização de seu acesso ao guarda-volumes, respeitados os procedimentos de segurança (senha e impressão digital).

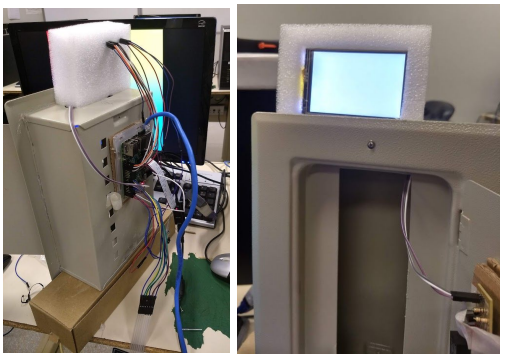


Fig. 1.Tentativa do sistema proposto. O protótipo final do projeto.

II Introdução

II.A Descrição do problema

Com muitas compras e itens cotidianos ao decorrer desse dia, pode-se ficar bastante inviável ou transtornante o indivíduo carregar seu inventário naquela ocasião. Mesmo o uso de mochilas ou sacolas podem não sanar as devidas necessidades, como por exemplo alguém sem carro que depois de um passeio vá para uma boate ou clube, onde não é permitida a entrada com vários tipos de objeto, ou mesmo a ida para um parque qualquer onde a pessoa sem carro não tem onde colocar vários objetos que carrega, nos quais obstruem a sua locomoção.

Seria bastante prático e útil para essas situações poder fazer o uso de um de vários guarda-volumes eletrônicos espalhados por sua cidade, bastando ter crédito previamente

adquirido e alguns cliques, para poder guardar seus objetos por um certo tempo.

A concepção do guarda-volumes eletrônico foi inspirada também no sistema da Amazon nos Estados Unidos no qual os produtos encomendados pelo site da empresa podem ser guardados em um "*Amazon Locker*" distribuídos pelas cidades, desde que haja espaço para guardar a encomenda [4]. Isso dá indício de que a ideia do guarda-volumes eletrônico é teoricamente funcional. Olhando do ponto de vista comercial, em que o projeto é visto como um produto em si, apesar de não ser cultural o uso de tal ferramenta em nosso meio, a ideia seria criar essa necessidade ao cliente, habilidade de marketing realizada por qualquer empresa ao lançar uma ideia nova no mercado.

Fig. 2. *Amazon Locker*. Nele você escolhe em qual lugar e o tamanho do cofre a receber seu produto.

II.B Objetivos

Tem-se como objetivo nesse projeto a construção de um guarda-volumes eletrônico que tenha:

- Aplicativo para o qual o usuário possa fazer o seu cadastro e pagamento do aluguel do guarda-volumes. Nele também é mostrado quanto tempo e créditos já foram gastos.
- Tela embarcada para visualização da interface pelo usuário, no qual será mostrado as opções de "alugue por até x horas" ou "Guarda-volumes cheio". No caso da primeira opção o usuário terá que entrar no sistema com o seu cadastro e credenciais para destravar a porta.
- Uma câmera para leitura de *QR Code*. Será utilizado para praticidade de uso pelo usuário. Ao se mostrar a imagem, já é detectada a conta do usuário no lugar em que está o armário.
- Um teclado para que o usuário digite sua senha de segurança.
- Acesso à internet para acessar os dados cadastrados do usuário do produto.
- Validação de acesso (abertura da porta) através da senha digitada e também com leitura da impressão digital.
- O acionamento de uma tranca elétrica (solenóide) para o destrancamento e trancamento da porta.
- Que o cofre seja de metal (importante para demonstrar que a implementação de hardware do protótipo pode ser factível em real escala).
- Toda vez que o usuário for utilizar ou reutilizar o guarda volumes, ele terá que recadastrar sua impressão digital, a fim de evitar situações como invalidação da digital cadastrada.

Uma ilustração do esquema pode-se vista no Apendice H.

III Desenvolvimento

III.A Descrição Inicial

Para o projeto final se teve inicialmente as seguintes metas:

1. Desenvolvimento do software de login de acesso do usuário ao guarda volumes;
2. Implantação de comunicação via servidor pelo sistema dos guarda-volumes com um servidor central.
3. Desenvolvimento de software para cadastro de contas (no computador do usuário) que faz a solicitação e cadastro de uma conta no servidor central;
4. Implementação de teclado numérico no terminal do guarda-volumes;
5. Implementação de leitor de impressão digital;
6. Implementação de câmera para leitura do QrCode;
7. Implementação e teste de tela integrada.

Assim para os primeiros passos do desenvolvimento do projeto foi necessário algumas configurações no Raspberry Pi.

O Raspberry Pi é um minicomputador de fácil interação.Somente é necessário ligá-lo a um monitor, ter um mouse e um teclado para usá-lo como se fosse um outro computador qualquer. Porém para o desenvolvimento da matéria onde iremos tentar construir e implementar um sistema embarcado, não é viável carregar tais periféricos para a configuração do raspberry.É muito mais prático utilizar outro computador para fazer as alterações a

“distância” deste minicomputador.Assim foram os passos seguidos:

1. Inicialmente o cartão SD foi formatado para que depois fosse instalado o SO Raspbian a partir do site oficial do Raspberry Pi[2].A versão do sistema operacional escolhido foi o “ Raspbian Buster with desktop”, uma versão adequada para utilizar um ambiente gráfico.Para a instalação foi utilizado o programa ETCHER[1].
2. Com o cartão inserido na Raspberry foi conectado no mesmo os periféricos: Tv(monitor) via HDMI, mouse e teclado via USB;
3. Ao ligar o sistema foi possível verificar no monitor a correta inicialização do sistema.Duas configurações foram modificadas , para que fosse possível usar o Raspberry de outro computador:as opções SSH e VNC são ligadas na aba de interfaces.O SSH é a opção que ativa o servidor (Secure Shell), que permite entrar no raspberry remotamente pela rede, o acesso é feito via linha de código, já o VNC o acesso é feito graficamente abrindo-se uma janela que simula a tela de visualização do sistema através do programa “Remmina”;
4. Pelo terminal do computador é feito os seguintes comandos:

```
$ if config  -- onde toda as configurações dos dispositivos conectados a rede aparecem

$ sudo apt-get install nmap

$ nmap -sV -p 22 192.168.0.11-255  -- faz un scan dos dispositivos conectados na rede que tem um número de IP similar ao número em negrito;

$ ssh pi@192.168.0.30  -- comando usado para ter o acesso remoto por linha de código à raspberry.O número em negrito é modificado pelo número de IP da raspberry encontrado no passo anterior;
```

Executando no computador pessoal o último comando , o cabeçalho do terminal muda para pi@raspberrypi:~$, o que significa que tudo digitado no terminal terá ação no sistema da Raspberry.

5- Para configurar o VNC é necessário os seguintes comandos, procedimento que só pode ser feito a partir dos passos anteriores:

```
$ sudo apt-get install tightvncserver   --- instalando o servidor VNC

$ vncserver :1 - geometry 1024x600 -depth 16 -pixelformat rgb565 -- exemplo de configuração do tamanho da tela a ser aberta
```

6 -Voltando o terminal para o sistema Linux do PC, instalou o VNC viewer para cliente

```
$ sudo apt-get install xtightvncviewer

$ vncviewer xxx.xxx.x.xxx:5901 --- Conectando o servidor VNC.O número em negrito é o número do IP da raspberry.
```

7 -Um problema foi encontrado ao tentar acessar remotamente o raspberry na UnB.A Raspberry não reconhece as redes Wi-fi disponíveis.Como proceder se o acesso remoto só é feito quando o computador e a raspberry são conectados à mesma rede de internet.A solução é conectar com um cabo de rede o computador com o minicomputador.Configurar o computador para o acesso “somente via cabo” nas configurações de rede e digitar no terminal:

```
$ ssh pi@raspberrypi.local
```

Através do comando anterior o IP da raspberry é encontrado e o acesso ao sistema da mesma poderá ser feita pelo mesmo comando ssh pi@<**IP_Rasp>**

III.B HARDWARE UTILIZADO (BOM)

Os equipamentos e materiais que foram adquiridos para o projeto:

- Quadro de energia para servir de armário.
- Placa Raspberry Pi 3 modelo B +.
- Cartão SD de 8 gb.
- Teclado matricial 4x4 de contato (barramento UART)
- Tranca elétrica solenoide (acionamento em 12V).
- Módulo relé (acionamento em 5V) [4]
- Módulo óptico de impressão digital JM-101B [6]
- Fonte de 12V;
- Tela LCD 3.5 polegadas;
- Módulo Câmera 5MP .

Fig. 3.Modelo de quadro de energia a ser utilizado no protótipo.

A tranca foi implementada, o acionamento se dá por um sinal (via software que será explanado na próxima seção) para um relê, armando-o. Ele faz a solenoide (tranca) ser alimentada por uma fonte de 12V, ligando-a.

O software do sistema foi implementado no Raspberry Pi, seu funcionamento será explanado na próxima seção. O acesso ao software da placa se dá por protocolo SSH. Já é possível fazer o cadastro de usuários em arquivo de dados e acesso por login para uso do 'guarda-volumes' pelo usuário remotamente.

O software para o sistema central foi desenvolvido para cadastro e gerenciamento de usuário em arquivo de dados em servidor, seu funcionamento será explanado na próxima seção.

O software do guarda-volumes foi implementado no Raspberry Pi, seu funcionamento será explanado na próxima seção. O acesso ao software da placa se dá por protocolo SSH. Nele é possível fazer o acesso básico do usuário pelo guarda-volumes, a verificação remota de usuário pelo servidor central, com validação posterior do leitor de impressão digital e acionamento da tranca.
O módulo óptico de impressão digital se comunica via UART ao *Raspberry pi*.

Fig. 4. Raspberry Pi 3, modelo B.

III. Software Desenvolvido

1. Teclado Matricial

O teclado matricial 4x4 foi escolhido como um ítem de projeto por ser acessível financeiramente e por ter muitas referências de uso e programação.Para o ponto de controle 2 o teclado foi testado a partir de um código em python.Como o teclado não possui um circuito interno, tudo o que o programa precisa fazer é buscar um jeito de realizar a leitura de maneira correta e precisa.

Simplificando o que o código faz é: setar os pinos correspondentes às colunas como outputs e com o valor "1"(*high*) .Os pinos correspondentes às linhas são configurados como inputs com pull-up resistors.(Todos como "1")

Os GPiOs configurados como outputs são setados como "0" (*low)* um de cada vez.Assim quando uma tecla é pressionada uma linha fica no estado *low* , assim saberemos qual coluna foi selecionada para também estar no estado *low,* já que apenas uma coluna pode estar em *low* em um certo instante de tempo.

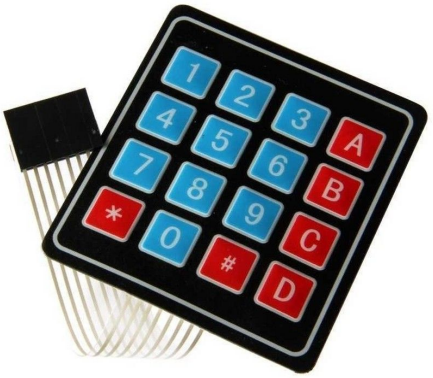


Fig. 5. Teclado matricial, 4x4.

O teclado foi implementado de maneira correta em c++ na raspiberry pi.Porém a dupla não teve tempo hábil de relacionar o código do teclado com o código do servidor, que serve como entrada ao sistema.

Uma ilustração do esquema pode-se vista no Apendice J.

2. Login Acionamento de tranca

Para o sistema do guarda-volumes, visando o uso do usuário, foi feito o programa 'cliente_terminal.c' que possibilita através de login na conta cadastrada previamente no servidor central, com comunicação via *socket* pela rede local e posteriormente cadastro da impressão digital e em seguida, caso validado, o acesso a abertura da porta do guarda-volumes (acionamento da tranca).

Para o correto funcionamento em sua execução, como os dispositivos conectados em rede local, são necessárias as seguintes entradas:

```
$./cliente_terminal <IP do Servidor> <Porta do servidor> <nome do usuario> <senha do usuario>
```

Se o usuário e senha inseridos forem válidos em seguida é solicitada a impressão digital do usuário. O mesmo dedo deve ser lido duas vezes. Caso procedimento tenha sucesso a tranca é liberada por 5 segundos.
Devendo estar no diretório do programa, via terminal no sistema Raspbian. O código do programa encontra-se no apêndice A.

Uma ilustração do algoritmo pode-se vista no Apendice K.

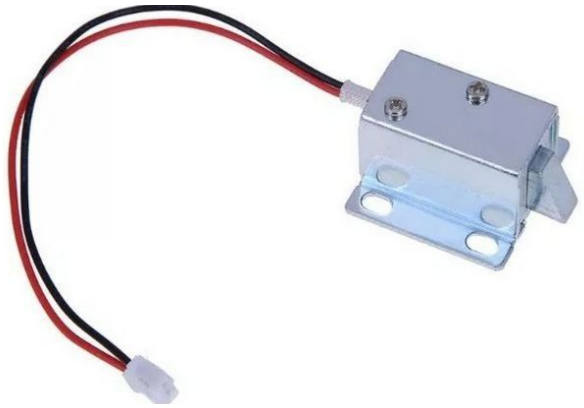


Fig. 6. Tranca solenoide..

O gerenciamento do módulo óptico de impressão digital foi feito pelo código do Apêndice D.

Fig. 7. Leitor óptico de impressão digital.

3. Software para gerenciamento dos dados no servidor

Para servidor web, foi feito o programa 'sistema_contas.c' que possibilita o cadastro de novos usuários, e leitura de usuários já previamente cadastrados no servidor central.

Primeiro é necessário abrir o arquivo 'sistema_contas' no servidor. Usa-se as entradas:

```
$./sistema_contas <Porta do servidor>
```

O arquivo 'portal_usuario' cria novos usuarios. Para o correto funcionamento em sua execução, são necessárias as seguintes entradas para ler algum usuário já cadastrado:

```
$./portal_usuario <IP do Servidor> <Porta do servidor> -l <nome do usuario>
```

Para cadastrar o usuario usa-se as entradas:

```
$./portal_usuario <IP do Servidor> <Porta do servidor> -m <nome do usuario> <senha> <numero de creditos>
```

devendo estar no diretório do programa, via terminal no sistema Raspbian. O código do programa encontra-se no apêndice B.

Para implementar as funções "adicionaUsuario(...)" e "leUsuario(...)" necessarias para os dois programas anteriores, foi feita a biblioteca 'gerencia_contas.h'.

O código do arquivo 'gerencia_contas.c' encontra-se no apêndice C.

Para a autenticação do login do usuário no terminal do guarda-volumes, foi implementado o programa 'servidor_sistema.c', que roda no servidor, onde há o arquivo com os dados dos usuários, 'usuarios.txt'. O programa do terminal 'cliente_terminal' envia os dados digitados pelo usuário a este, via *socket*, que faz a verificação dos dados e envia uma resposta ao terminal.

O código do arquivo 'servidor_sistema.c' encontra-se no apêndice F.

Os arquivos supracitados encontram-se em [3]

4. Tela LCD

Uma pequena tela LCD foi implementada usando os seguintes passos:

- Na pasta Download do sistema operacional instalado no Raspberry Pi foi baixado o driver presente no endereço :

  wget http://en.kedei.net/raspberry/v6_1/LCD_show_v6_1_3.tar.gz

- Instalação do arquivo dentro da pasta baixada : `sudo ./LCD35_v`
- Após a instalação o Raspberry fez um reboot automaticamente e foi necessário digitar o seguinte comando no terminal para alterar mais uma configuração interna: `raspi-config`
- A configuração interna modificada foi dentro da aba " Advanced Options" , onde a opção " A6 GL DRIVER(FULL KMS)" foi habilitada;
- A última modificação feita foi dentro do arquivo de texto "config.txt" dentro da pasta boot.Este de arquivo de texto é lido durante a inicialização do sistema da raspberry pi, as linhas 28, 29 e 48 foram modificadas respectivamente pelas linhas:

  ```
  hdmi_group=2
  hdmi_mode=35
  dtparam=spi=off
  ```

E ao final do arquivo foram adicionadas as seguintes linhas de instrução:

```
hdmi_force_hotplug=1

gpu_mem=32

start_x=0
```

```
enable_uart=1

dtoverlay=w1-gpio
```

Após o sistema ser reiniciado a tela já ligava normalmente, aparecendo a imagem inicial do desktop do sistema operacional raspbian.

E. Câmera

A instalação da câmera foi feita de maneira bem mais rápida e convencional que a tela.A câmera tem espaço reservado na raspberry B +, sem ocupar nenhum pino ou porta USB. A única modificação a ser feita é habilitar a câmera dentro das configurações do raspberry dentro da opção 5- iNTERFACING OPTIONS .

Comandos básicos foram testados para averiguar o funcionamento da câmera como:

`raspistill -o minha_foto.png` - Para gerar uma foto atravéz da câmera

`raspivid -o nome_do_video.h264 -t 10000` - gravar um vídeo de 5s de duração

F. Geração de Qrcode

Efetivamente a dupla utilizou um código no qual transforma um texto no formato de Qrcode .A intenção inicial era que para cada usuário seria gerado um Qrcode personalizado utilizando para a sua construção os dados pessoais do mesmo.

Faltou a dupla conseguir transformar a imagem impressa no terminal em imagem, algo facilmente achado como referência em outras linguagens, como JAVA.

IV Resultados

IV.A - Entrega final

Foi mostrado o projeto ao professor dia para apresentação final. São ressaltados os seguintes detalhes:

1. Sistema final foi mostrado e é mostrado na figura 8
2. O programa que verifica o usuário cadastrado no servidor funcionou corretamente, embora a tranca, por problemas físicos, não parasse de estar acionada.

A versão final do protótipo ficou sem vários dos itens pretendidos no projetos

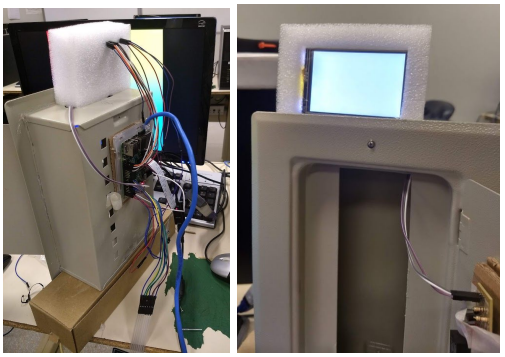
3. 

Fig. 8.Tentativa do sistema proposto. O protótipo final do projeto.

IV.B - Problemas enfrentados

Durante a apresentação final ocorreram vários problemas no projeto, onde a maioria deles é justificada pela falta de tempo hábil dos membros do grupo, como recursos em falta na versão final do protótipo.

Fatos importantes ocorridos:

1. Dificuldade dos membros do grupo em trabalhar com a carcaça para o guarda-volumes (quadro de energia) e fazer os acabamentos.
2. A Raspberry pi utilizada pelo grupo teve problemas durante o período de realização do projeto.Aconteceu algumas vezes em que ao se dar o comando “sudo upgrade”, o sistema operacional “crashava”, e não era possível reiniciar o boot do sistema.O cartão precisava ser formatado para reinstalar o Raspian.
3. Não foi possível implementar uma interface gráfica para o usuario do guarda-volumes
4. Não foi possível fazer a implementação do leitor de impressão digital. A documentação presente em seu manual é pobre para um sensor com modos de funcionamento bastante complexos.
5. O acionamento da tranca seu de forma errada e anômala durante a apresentação, onde a tranca não parava de ser energizada. Esse problema nunca foi detectado antes em testes anteriores.

V Conclusão

Na implementação de um guarda-volumes é sem duvidas necessário, além da integração de vários componentes de hardware distintos, uma conexão com servidor dedicado é fundamental. A confiabilidade dos componentes é fundamental e também da conexão entre servidor/terminal.De fato a equipe nao conseguiu realizar o projeto com sucesso, por vários erros isolados, que no total não deixou que o projeto funcionasse de acordo com a proposta inicial da equipe.

Revisão Bibliográfica

[1] Balena ETCHER, <https://www.balena.io/etcher/l>. Acesso em 1 de outubro de 2019.

[2] Raspberry Pi <https://www.raspberrypi.org/>. Acesso em 1 de outubro de 2019.

[3] Repositório <https://github.com/gabrielbopi/SO_Embarcados/tree/master/3_Trabalho/Arquivoso>. Acesso em 7 de dezembro de 2019.

[4] Módulo Relê, <https://www.huinfinito.com.br/modulos/988-modulo-rele-5v1canal.html>. Acesso em 1 de novembro de 2019.

[5] Python library for ZFM fingerprint sensors. <https://github.com/bastianraschke/pyfingerprint>. Acesso em 1 de novembro de 2019.

[6] JM-101 Optical Fingerprint Module User Manual. <http://21st-century-spring.kr/wordpress/?page_id=794>. Acesso em 1 de novembro de 2019.

APÊNDICE

Seguem os códigos citados na seção IV.

A. Programa ‘cliente_terminal.c’. O código pode ser compilado pelo GCC e necessita da biblioteca ‘gerencia_contas.h’ e ‘gpio_sysfs.h’ no mesmo diretório. Também é necessario o arquivo ‘/home/pi/Desktop/Codigo/fingerprint_enroll.py’, e consequentemente o Python instalado no sistema. É necessária conexão de rede local com o servidor principal rodando ‘’.

```
       #include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <sys/un.h>
#include <unistd.h>
#include "gerencia_contas.h"
#include <time.h>
#include "gpio_sysfs.h"

#define TEMPO_CREDITOS 5
#define entrada_usuario argv[3]
#define entrada_senha argv[4]

struct usuario usuario_corrente;
time_t time_1, time_2;


int recolheDigital(void);
int abreTranca(void);
void contagemCreditos(int sig);

int main(int argc, char* const argv[]){
       int socket_id;
       struct sockaddr_in servidorAddr;
       int length;
       unsigned short servidorPorta;
       char *IP_Servidor;
       char *mensagem;
       int resposta;
       char nome_recolhe[30], senha_recolhe[30];

       const int creditos_minimos = 5;
       int polegarAutenticado;

       if (argc < 3){
              printf("Escolha uma opcao valida.\n");
              printf("Para utilizar o programa escreva no terminal:\n./cliente_terminal
<IP do Servidor> <Porta do servidor> <nome do usuario> <senha do usuario>\n");
              return -1;
       }

       strcpy(usuario_corrente.nome, entrada_usuario);
       strcpy(usuario_corrente.senha, entrada_senha);

       IP_Servidor = argv[1];
       servidorPorta = atoi(argv[2]);
       //mensagem = argv[3];
```

```
	// Abrindo o socket para o cliente
	socket_id = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
	if(socket_id < 0){
		fprintf(stderr, "Erro na criacao do Soquete!\n");
		exit(0);
	}

	// Conectando o socket ao IP "IP_Servidor" pela porta "servidorPorta"
	memset(&servidorAddr, 0, sizeof(servidorAddr));
	servidorAddr.sin_family = AF_INET;
	servidorAddr.sin_addr.s_addr = inet_addr(IP_Servidor);
	servidorAddr.sin_port =  htons(servidorPorta);
	if(connect(socket_id, (struct sockaddr *) &servidorAddr, sizeof(servidorAddr)) <
0){
		fprintf(stderr, "Erro na conexao!\n");
		exit(0);
	}

	// Mandando mensagem ao servidor
	length = sizeof(struct usuario);
	//write(socket_id, &length, sizeof(length));
	write(socket_id, &usuario_corrente, length);
	read(socket_id, &resposta, sizeof(resposta));
	//fprintf(stderr, "%d bytes.", length);
	read(socket_id, &usuario_corrente, length);
	// Fechando o socket local
	close(socket_id);

	if(resposta == 1){
		printf("Usuario reconhecido. Bem vindo!\nUsuario: %s, senha:%s,
creditos:%d\n",
				usuario_corrente.nome, usuario_corrente.senha,
usuario_corrente.creditos);
		polegarAutenticado = recolheDigital()/256;
		printf("Resposta: %d\n", polegarAutenticado);

		if(polegarAutenticado == 10) {
				abreTranca();
				printf("Abre-te Sesamo!\n");
				signal(SIGALRM, contagemCreditos);
				time_1 = time(NULL);
				alarm(TEMPO_CREDITOS);
			while(1){
				printf("Para abrir o guarda-volumes, digite seu usuario:\n");
				while(1){
					scanf("%s", nome_recolhe);
					printf("Usuario correne: %s, nome_recolhe:%s",
usuario_corrente.nome, nome_recolhe);

					if(strcmp(usuario_corrente.nome,nome_recolhe) == 0){
						while(1){
							printf("Agora a senha:\n");
							scanf("%s", senha_recolhe);

if(strcmp(usuario_corrente.senha,senha_recolhe) == 0){
								printf("Abre-te Sesamo!\n");
								printf("Obrigado por utilizar no
serviço!\n");
								abreTranca();
								return 0;
							}
						printf("\nSenha incorreta.  Digite
```

```
novamente\n");
                                                }
                                        }
                                        printf("\nNome incorreto. Digite novamente\n");
                                }

                        }
                }
        }
        else if(resposta == 2)     {printf("Senha incorreta!\n");}
        else if(resposta == 0)     {printf("Usuario escrito escrito errado ou
inexistente.\n");}
        else {printf("Erro no servidor.\n");}

        return 0;
}

void contagemCreditos(int sig){
        time_2 = time(NULL);
        printf("Creditos restantes: %d\n", usuario_corrente.creditos--);
        printf("Tempo de uso do guarda volumes: %4.0f s\n", difftime(time_2, time_1));
        printf("Tempo restante: %d s\n", (usuario_corrente.creditos*TEMPO_CREDITOS));
        sleep(TEMPO_CREDITOS);
        if(usuario_corrente.creditos < 0){
                printf("Objeto retido. Creditos insuficientes para continuar servico.\n");
                while(1);
        }
        alarm(TEMPO_CREDITOS);
}


int recolheDigital(void){
        int resposta;

        resposta = system("python /home/pi/Desktop/Codigo/fingerprint_enroll.py");
        printf("Resposta: %d\n", resposta);
        return resposta;
}

int abreTranca(void){
        int pin=4;
        if(setGPIO_Out(pin)){
                printf("Erro ao tentar abrir a porta...(1)\n");
                return -1;
        }
        if (GPIO_Write(pin,1)){
                printf("Erro ao tentar abrir a porta...(2)\n");
                return 1;
                }
        printf("Tranca abre...\n");
        sleep(6);
        if (GPIO_Write(pin,0)){
                printf("Erro ao tentar desligar a tranca...(1)\n");
                return 3;
                }
        if(unsetGPIO(pin)){
                printf("Erro ao tentar desligar a tranca...(2)\n");
                return 2;
        }
        return 0;
}
```

B. Programa ‘sistema_contas.c’. O código pode ser compilado pelo GCC e necessita da biblioteca gerencia_contas.h no mesmo diretório.

```
//Servidor central para gerenciamento dos usuarios
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <string.h>
#include <signal.h>
#include <sys/socket.h>
#include <sys/un.h>
#include "gerencia_contas.h"
#define ARQUIVO_CONTAS        "usuarios.txt"

#define arquivo_dados "usuarios.txt"
#define entrada_usuario argv[3]
#define entrada_senha argv[4]
#define entrada_creditos argv[5]

/*
struct usuario
{
        int id;
        char nome[30], senha[30];
        int creditos;
};
*/

int socket_id;
void sigint_handler(int signum);
void procuraUsuario(int client_socket);
void end_server(void);

int main (int argc, char* const argv[]){
        unsigned short servidorPorta;
        struct sockaddr_in servidorAddr;

        if (argc < 2) {
                printf("Escolha uma opcao valida.\n");
                printf("Para utilizar o programa escreva no terminal:\n$ ./sistema_contas <Porta do servidor>\n");
                return -1;
        }
        // Definindo o tratamento de SIGINT
        servidorPorta = atoi(argv[1]);
        signal(SIGINT, sigint_handler);

        // Abrindo o socket local
        socket_id = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
        if(socket_id < 0)
        {
                fprintf(stderr, "Erro na criacao do Soquete!\n");
```

```
		exit(-1);
	}
	// Ligando o socket a porta "servidorPorta"
	memset(&servidorAddr, 0, sizeof(servidorAddr));
	servidorAddr.sin_family = AF_INET;
	servidorAddr.sin_addr.s_addr = htonl(INADDR_ANY);
	servidorAddr.sin_port = htons(servidorPorta);
	if(bind(socket_id, (struct sockaddr *) &servidorAddr, sizeof(servidorAddr)) < 0){
		fprintf(stderr, "Erro na ligacao!\n");
		exit(-1);
	}
	// Tornando o socket passivo (para virar um servidor)
	if(listen(socket_id,10) < 0){
		fprintf(stderr, "Erro!\n");
		exit(0);
	}

	while(1){
		int socketCliente;
		struct sockaddr_in clienteAddr;
		unsigned int clienteLength;
		// Aguardando a conexao de um cliente
		clienteLength = sizeof(clienteAddr);
		if((socketCliente = accept(socket_id, (struct sockaddr *) &clienteAddr, &clienteLength)) < 0)
		{	fprintf(stderr, "Falha no accept().\n");}

		fprintf(stderr, "Sistema para autenticacao de usuarios para o guarda-volumes.\nConexao do Cliente
%s\n", inet_ntoa(clienteAddr.sin_addr));
		// Tratando comunicacao com o cliente
		procuraUsuario(socketCliente);
		// Fechando a conexao com o cliente
		close(socketCliente);
	}

	return 0;
}

void sigint_handler(int signum){
	fprintf(stderr, "\nRecebido o sinal CTRL+C... vamos desligar o servidor!\n");
	end_server();
}

void procuraUsuario(int client_socket){
	int length, nome_valido, senha_valida;
	int resposta;
	char pedido;
	char* text;
	struct usuario usuario_solicitado, usuario_corrente;

	printf("Aguardando alguma solicitacao de usuario...\n");
	length = sizeof(struct usuario);
	read(client_socket, &pedido, sizeof(char));
	read(client_socket, &usuario_solicitado, length);
```

```
                switch(pedido){
                        case 'l':
                                usuario_corrente = leUsuario(arquivo_dados,usuario_solicitado.nome);
                                resposta = usuario_corrente.id;
                                printf("ID: %d, usuario: %s, senha:%s, creditos:%d\n", usuario_corrente.id,
usuario_corrente.nome, usuario_corrente.senha, usuario_corrente.creditos);
                        break;
                        case 'm':
                                resposta =
adicionaUsuario(arquivo_dados,usuario_solicitado.nome,usuario_solicitado.senha, usuario_solicitado.creditos);
                                if (resposta != 0) {printf("Erro no cadastro!\n");}
                        break;
                        defaut:
                        printf("Escolha uma opcao valida.\n");
                        printf("Para utilizar o programa escreva:\nPara ler algum usuario cadastrado:
./sistema_contas -l usuarios.txt <nome do usuario>\nPara cadastrar o usuario: ./sistema_contas -m usuarios.txt <nome
do usuario> <senha> <numero de creditos>\n");
                        }
        write(client_socket, &resposta, sizeof(resposta));
        write(client_socket, &usuario_corrente, length);
}

void end_server(void){
        //Fechando o socket local
        close(socket_id);
        exit(0);
}
```

C. Programa ‘portal_usuario.c’. O código pode ser compilado pelo GCC e necessita da biblioteca gerencia_contas.h no mesmo diretório.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <sys/un.h>
#include <unistd.h>
#include "gerencia_contas.h"
#define entrada_usuario argv[3]
#define entrada_senha argv[4]

int main(int argc, char* const argv[]){
       int socket_id;
       struct sockaddr_in servidorAddr;
       int length;
       unsigned short servidorPorta;
       char *IP_Servidor;
       char *mensagem;
       int resposta;

       const int creditos_minimos = 5;
       struct usuario usuario_corrente;
       int polegarAutenticado;
       char pedido;
```

```
if (argc < 3) {
		printf("Escolha uma opcao valida.\n");
		printf("Para utilizar o programa escreva no terminal:\n- Para ler algum
usuario cadastrado: ./portal_usuario <IP do Servidor> <Porta do servidor> -l <nome do
usuario>\n- Para cadastrar o usuario: ./portal_usuario <IP do Servidor> <Porta do
servidor> -m <nome do usuario> <senha> <numero de creditos>\n");
		return -1;
	}
	if(argv[3][0]=='-'){
		switch(argv[3][1]){
			case 'l':
					//usuario_corrente =
leUsuario(arquivo_dados,entrada_usuario);
				pedido = 'l';
				printf("ID: %d, usuario: %s, senha:%s, creditos:%d\n",
usuario_corrente.id, usuario_corrente.nome, usuario_corrente.senha,
usuario_corrente.creditos);
			break;
			case 'm':
				if (argc < 5) {
					printf("Escolha uma opcao valida.\n");
					printf("Para utilizar o programa escreva:\nPara ler
algum usuario cadastrado: ./sistema_contas -l usuarios.txt <nome do usuario>\nPara
cadastrar o usuario: ./sistema_contas -m usuarios.txt <nome do usuario> <senha> <numero
de creditos>\n");
					return -1;
				}
				pedido = 'm';
			break;
			defaut:
			printf("Escolha uma opcao valida.\n");
			printf("Para utilizar o programa escreva no terminal:\n- Para ler
algum usuario cadastrado: ./portal_usuario <IP do Servidor> <Porta do servidor> -l <nome
do usuario>\n- Para cadastrar o usuario: ./portal_usuario <IP do Servidor> <Porta do
servidor> -m <nome do usuario> <senha> <numero de creditos>\n");

			}
		}

	else {
		printf("Para utilizar o programa escreva no terminal:\n- Para ler algum
usuario cadastrado: ./portal_usuario <IP do Servidor> <Porta do servidor> -l <nome do
usuario>\n- Para cadastrar o usuario: ./portal_usuario <IP do Servidor> <Porta do
servidor> -m <nome do usuario> <senha> <numero de creditos>\n");
	}

	strcpy(usuario_corrente.nome, entrada_usuario);
	strcpy(usuario_corrente.senha, entrada_senha);

	IP_Servidor = argv[1];
	servidorPorta = atoi(argv[2]);

	// Abrindo o socket para o cliente
	socket_id = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
	if(socket_id < 0){
		fprintf(stderr, "Erro na criacao do Soquete!\n");
		exit(0);
	}

	// Conectando o socket ao IP "IP_Servidor" pela porta "servidorPorta"
	memset(&servidorAddr, 0, sizeof(servidorAddr));
```

```
	servidorAddr.sin_family = AF_INET;
	servidorAddr.sin_addr.s_addr = inet_addr(IP_Servidor);
	servidorAddr.sin_port =  htons(servidorPorta);
	if(connect(socket_id, (struct sockaddr *) &servidorAddr, sizeof(servidorAddr)) <
0){
		fprintf(stderr, "Erro na conexao!\n");
		exit(0);
	}

	// Mandando mensagem ao servidor
	length = sizeof(struct usuario);
	//write(socket_id, &length, sizeof(length));
	write(socket_id, &pedido, sizeof(char));
	write(socket_id, &usuario_corrente, length);
	read(socket_id, &resposta, sizeof(resposta));
	//fprintf(stderr, "%d bytes.", length);
	read(socket_id, &usuario_corrente, length);
	// Fechando o socket local
	close(socket_id);


	switch(pedido){
			case 'l':
				if(resposta == -1)  {printf("Usuario nao existente
anteriormente no sistema.\n");}
				else   {printf("ID: %d, usuario: %s, senha:%s,
creditos:%d\n", usuario_corrente.id, usuario_corrente.nome, usuario_corrente.senha,
usuario_corrente.creditos);}
			break;

			case 'm':
				if (resposta == 0)  {printf("Usuario cadastrado com
sucesso!\n");}
				else if(resposta == -1)    {printf("Erro. Esse usuario ja
existe.\n" );}
				else   {printf("Erro no cadastro de usuario!\n");}

			break;
			}

	return 0;
}
```

D. Biblioteca ‘gerencia_contas.c’. O código pode ser compilado como objeto pelo GCC e necessita do arquivo gerencia_contas.h no mesmo diretório.

```
		#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include "gerencia_contas.h"

/*struct usuario
{
	int id;
	char nome[30], senha[30];
	int creditos;
};*/
```

```
struct usuario leUsuario(char *nome_arquivo, char *nome_usuario){
    int fp;
       char i;
       struct usuario usuario_corrente;

       int offset1, offset2;
       usuario_corrente.id = 0;

    fp = open(nome_arquivo,O_RDONLY | O_CREAT, S_IRWXU);
    if(fp == -1){
              printf("Erro ao abrir o arquivo!");
              exit(EXIT_FAILURE);
    }

    offset2 = sizeof(usuario_corrente.senha) + sizeof(usuario_corrente.creditos);

       while(read(fp, &i, sizeof(char)) != 0) {
           offset1 = usuario_corrente.id*sizeof(struct usuario) + sizeof(int);
           lseek(fp, offset1, SEEK_SET);
           //printf("usuario_corrente.nome:%s nome_usuario:%s\n", usuario_corrente.nome,
nome_usuario);
           read(fp, usuario_corrente.nome, 30 * sizeof(char));
              if (strcmp(usuario_corrente.nome, nome_usuario) == 0){
                  read(fp, usuario_corrente.senha, 30 * sizeof(char));
                  read(fp, &usuario_corrente.creditos, sizeof(int));

                     return usuario_corrente;
              }
              else{
                     lseek(fp, offset2, SEEK_CUR);
                     usuario_corrente.id++;
              }
       }
       usuario_corrente.id = -1;
       strcpy(usuario_corrente.nome,"");
       strcpy(usuario_corrente.senha,"");
       usuario_corrente.creditos = 0;


       printf("Usuario nao existente anteriormente no sistema.\n");
       return usuario_corrente;
}

int adicionaUsuario(char *nome_arquivo, char *nome_usuario,char *senha_usuario, int
creditos_usuario){
    int fp;
       struct usuario usuario_corrente;

       usuario_corrente = leUsuario(nome_arquivo, nome_usuario);
       if(strcmp(usuario_corrente.nome, nome_usuario) == 0){
              printf("Erro. Esse usuario ja existe.\n" );
              return -1;
       }
       else{
              strcpy(usuario_corrente.nome,nome_usuario);
              strcpy(usuario_corrente.senha,senha_usuario);
              usuario_corrente.creditos = creditos_usuario;

              fp = open(nome_arquivo, O_WRONLY | O_APPEND | O_CREAT, S_IRWXU);
       if(fp == -1){
              printf("Erro ao abrir o arquivo!");
```

```
            exit(EXIT_FAILURE);
        }

        if(write(fp, &usuario_corrente, sizeof(struct usuario)) != -1){
            printf("Usuario cadastrado com sucesso!\n");
        }
      else{
            printf("Erro no cadastro de usuario!\n");
            return -2;
      }
            return 0;
      }
}
```

E. Código do teclado Matricial 4X4 em C

```
#include <wiringPi.h>
#include <stdio.h>

#define ROWS 4

#define COLS 4

char pressedKey = '\0';

int rowPins[ROWS] = {1, 4, 5, 6};

int colPins[COLS] = {12, 3, 2, 0};


char keys[ROWS][COLS] = {

{'1', '2', '3', 'A'},

 {'4', '5', '6', 'B'},

{'7', '8', '9', 'C'},

 {'*', '0', '#', 'D'} };

void init_keypad()

{

   for (int c = 0; c < COLS; c++)

   {

      pinMode(colPins[c], OUTPUT);
```

```
      digitalWrite(colPins[c], HIGH);
   }


   for (int r = 0; r < ROWS; r++)
   {
      pinMode(rowPins[0], INPUT);
      digitalWrite(rowPins[r], PUD_UP);
   }
}
int findLowRow()
{
   for (int r = 0; r < ROWS; r++)
   {
      if (digitalRead(rowPins[r]) == LOW)
 return r;
   }   return -1;
}
char get_key()
{   int rowIndex;
   for (int c = 0; c < COLS; c++) {
      digitalWrite(colPins[c], LOW);
        rowIndex = findLowRow();
      if (rowIndex > -1)
      {         if (!pressedKey)
            pressedKey = keys[rowIndex][c];
         return pressedKey;
      }
      digitalWrite(colPins[c], HIGH);
```

```
}   pressedKey = '\0';

   return pressedKey;

}

int main(void)

{   wiringPiSetup();

   init_keypad();

   while(1)

   {      char x = get_key();

      if (x)

         printf("pressed: %c\n", x);

      else

         printf("no key pressed\n");

      delay(250);

   }   return 0;

}
```

F. Código do '/home/pi/Desktop/Codigo/fingerprint_enroll.py' em Python. Ele é proveniente do repositório em [5]

```
"""
PyFingerprint
Copyright (C) 2015 Bastian Raschke <bastian.raschke@posteo.de>
All rights reserved.

"""

import time
from pyfingerprint.pyfingerprint import PyFingerprint


## Enrolls new finger
##

## Tries to initialize the sensor
try:
    f = PyFingerprint('/dev/ttyS0', 57600, 0xFFFFFFFF, 0x00000000)

    if ( f.verifyPassword() == False ):
        raise ValueError('The given fingerprint sensor password is wrong!')
```

```
except Exception as e:
    print('The fingerprint sensor could not be initialized!')
    print('Exception message: ' + str(e))
    exit(1)

## Gets some sensor information
print('Currently used templates: ' + str(f.getTemplateCount()) +'/'+
str(f.getStorageCapacity()))

## Tries to enroll new finger
try:
    print('Waiting for finger...')

    ## Wait that finger is read
    while ( f.readImage() == False ):
        pass

    ## Converts read image to characteristics and stores it in charbuffer 1
    f.convertImage(0x01)

    ## Checks if finger is already enrolled
    result = f.searchTemplate()
    positionNumber = result[0]

    if ( positionNumber >= 0 ):
        print('Template already exists at position #' + str(positionNumber))
        exit(0)

    print('Remove finger...')
    time.sleep(2)

    print('Waiting for same finger again...')

    ## Wait that finger is read again
    while ( f.readImage() == False ):
        pass

    ## Converts read image to characteristics and stores it in charbuffer 2
    f.convertImage(0x02)

    ## Compares the charbuffers
    if ( f.compareCharacteristics() == 0 ):
        raise Exception('Fingers do not match')

    ## Creates a template
    f.createTemplate()

    ## Saves template at new position number
    positionNumber = f.storeTemplate()
    print('Finger enrolled successfully!')
    print('New template position #' + str(positionNumber))
    exit(10)

except Exception as e:
    print('Operation failed!')
    print('Exception message: ' + str(e))
    exit(1)
```

G. Programa ‘servidor_sistema.c’. O código pode ser compilado pelo GCC e necessita da biblioteca gerencia_contas.h no mesmo diretório.

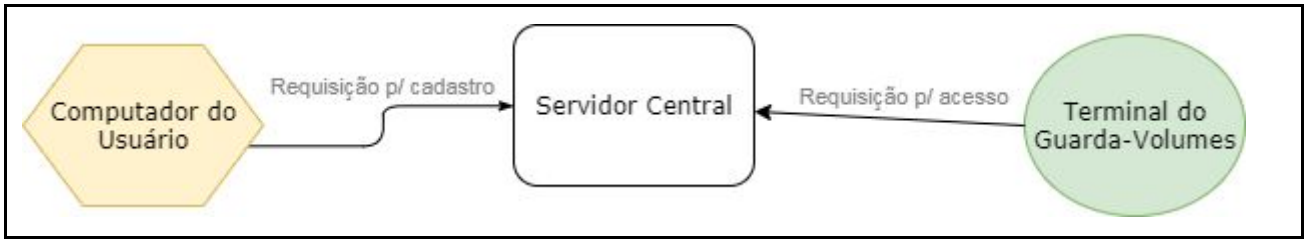
```
              //Servidor central para gerenciamento dos usuarios
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <string.h>
#include <signal.h>
#include <sys/socket.h>
#include <sys/un.h>
#include "gerencia_contas.h"
#define ARQUIVO_CONTAS     "usuarios.txt"

int socket_id;
void sigint_handler(int signum);
void procuraUsuario(int client_socket);
void end_server(void);

int main (int argc, char* const argv[]){
       unsigned short servidorPorta;
       struct sockaddr_in servidorAddr;

       if (argc < 2) {
              printf("Escolha uma opcao valida.\n");
              printf("Para utilizar o programa escreva no terminal:\n$
./servidor_sistema <Porta do servidor>\n");
              return -1;
       }
       // Definindo o tratamento de SIGINT
       servidorPorta = atoi(argv[1]);
       signal(SIGINT, sigint_handler);

       // Abrindo o socket local
       socket_id = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
       if(socket_id < 0)
       {
              fprintf(stderr, "Erro na criacao do Soquete!\n");
              exit(-1);
       }
       // Ligando o socket a porta "servidorPorta"
       memset(&servidorAddr, 0, sizeof(servidorAddr));
       servidorAddr.sin_family = AF_INET;
       servidorAddr.sin_addr.s_addr = htonl(INADDR_ANY);
       servidorAddr.sin_port = htons(servidorPorta);
       if(bind(socket_id, (struct sockaddr *) &servidorAddr, sizeof(servidorAddr)) < 0){
              fprintf(stderr, "Erro na ligacao!\n");
              exit(-1);
       }
       // Tornando o socket passivo (para virar um servidor)
       if(listen(socket_id,10) < 0){
              fprintf(stderr, "Erro!\n");
              exit(0);
       }

       while(1){
              int socketCliente;
              struct sockaddr_in clienteAddr;
              unsigned int clienteLength;
              // Aguardando a conexao de um cliente
              clienteLength = sizeof(clienteAddr);
```

```
            if((socketCliente = accept(socket_id, (struct sockaddr *) &clienteAddr,
&clienteLength)) < 0)
            {      fprintf(stderr, "Falha no accept().\n");}

            fprintf(stderr, "Sistema para autenticacao de usuarios para o
guarda-volumes.\nConexao do Cliente %s\n", inet_ntoa(clienteAddr.sin_addr));
            // Tratando comunicacao com o cliente
            procuraUsuario(socketCliente);
            // Fechando a conexao com o cliente
            close(socketCliente);
      }

      return 0;
}

void sigint_handler(int signum){
      fprintf(stderr, "\nRecebido o sinal CTRL+C... vamos desligar o servidor!\n");
      end_server();
}

void procuraUsuario(int client_socket){
      int length, nome_valido, senha_valida;
      char* text;
      struct usuario usuario_solicitado, usuario_corrente;
      int resposta = 0;

      printf("Aguardando alguma solicitacao de usuario...\n");
      length = sizeof(struct usuario);
      read(client_socket, &usuario_solicitado, length);

      usuario_corrente = leUsuario(ARQUIVO_CONTAS,usuario_solicitado.nome);
      nome_valido = strcmp(usuario_solicitado.nome, usuario_corrente.nome);
      senha_valida = strcmp(usuario_solicitado.senha, usuario_corrente.senha);

      if(nome_valido == 0){
            if(senha_valida == 0){
                  resposta = 1;
                  printf("ID: %d, usuario: %s, senha:%s, creditos:%d\n",
                  usuario_corrente.id, usuario_corrente.nome, usuario_corrente.senha,
usuario_corrente.creditos);
                  write(client_socket, &resposta, sizeof(resposta));
                  write(client_socket, &usuario_corrente, length);
            }else{
                  resposta = 2;
                  printf("ID: %d, usuario: %s\nSenha incorreta!\n",
                        usuario_corrente.id, usuario_corrente.nome);
            }
      }else{
            resposta = 0;
            printf("Usuario nao encontrado...\n");
      }
}

void end_server(void){
      //Fechando o socket local
      close(socket_id);
      exit(0);
}
```

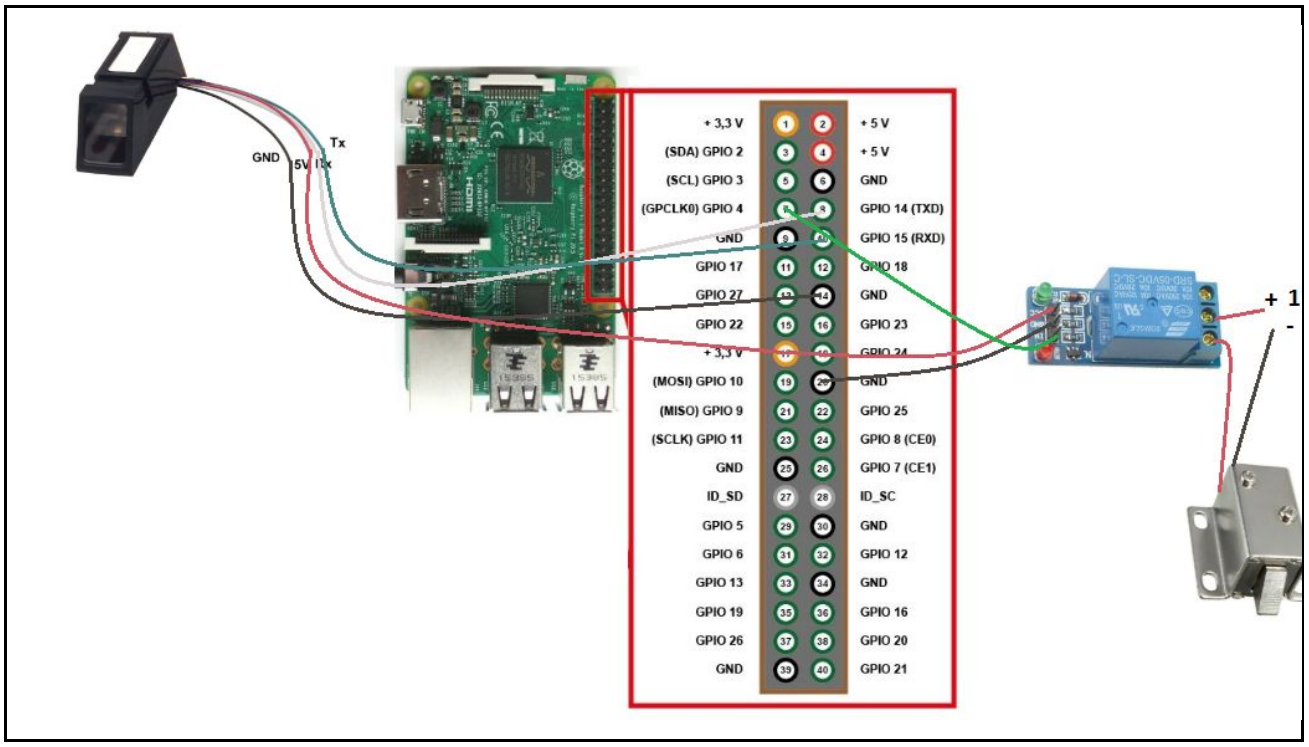
H -Programa gerador de Qrcode:

```
// gerador de qrcode
#include <cstdint>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <string>
#include <vector>
#include "BitBuffer.hpp"
#include "QrCode.hpp"

using std::uint8_t;
using qrcodegen::QrCode;
using qrcodegen::QrSegment;


static void doBasicDemo();
static void printQr(const QrCode &qr);


int main() {
	doBasicDemo();
	return EXIT_SUCCESS;

static void doBasicDemo() {
	const char *text = "Elisa";              // User-supplied text
	const QrCode::Ecc errCorLvl = QrCode::Ecc::LOW;  // Error correction level

	// Make and print the QR Code symbol
	const QrCode qr = QrCode::encodeText(text, errCorLvl);
	printQr(qr);
	std::cout << qr.toSvgString(4) << std::endl;
}


static void printQr(const QrCode &qr) {
	int border = 4;
	for (int y = -border; y < qr.getSize() + border; y++) {
		for (int x = -border; x < qr.getSize() + border; x++) {
			std::cout << (qr.getModule(x, y) ? "##" : "  ");
		}
		std::cout << std::endl;
	}
	std::cout << std::endl;
}
```

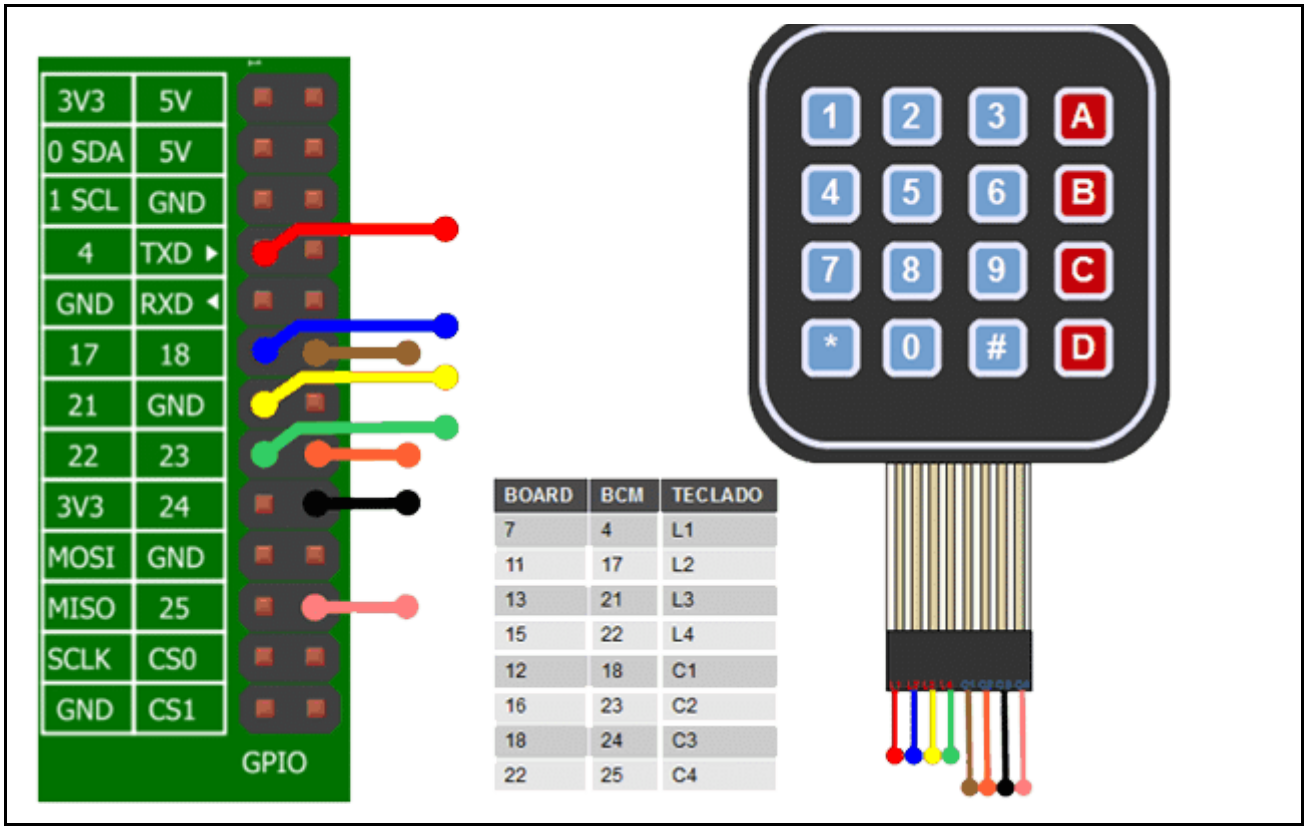
I- Diagrama mostrando esquema de ligação usuário/terminal/servidor.



J. Esquemático mostrando ligações entre alguns componentes (não foi considerada a tela LCD, esta usa os pinos SPI).



K.Esquemático mostrando ligações entre o teclado matricial.



BOARD	BCM	TECLADO
7	4	L1
11	17	L2
13	21	L3
15	22	L4
12	18	C1
16	23	C2
18	24	C3
22	25	C4

l. Diagrama ilustrando o algoritmo do programa do terminal do guarda-volumes..