Ponto de Controle 2 - 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 parcial do projeto no segundo ponto de controle.**
Palavras-Chave—guarda-volumes; praticidade; segurança; Raspberry Pi

I. Introdução

Para o segundo ponto de controle, era esperado o desenvolvimento de um protótipo funcional do projeto, utilizando as ferramentas mais básicas da placa de desenvolvimento, com bibliotecas prontas. Orientando-se por esse foco, tentou-se fazer a lista de requisitos e a aquisição dos materiais (hardware) necessários para todo o projeto, além desenvolvimento e integração básica dos componentes em que os membros do grupo já tinham acesso/posse.

Fig. 1.Modelo de quadro de energia a ser utilizado no protótipo.

II. Objetivos e Desenvolvimento

Para este ponto de controle foi pedido para que os alunos testassem implementar cada requisito do sistema em blocos funcionais, assim temos:

- Software de login de acesso do usuário ao guarda volumes;
- Teste de câmera para leitura do QrCode;
- Teste de teclado para inserção da senha de acesso;
- Implementação do leitor de digital;
- Teste e implementação da tranca elétrica ao sistema;
- Teste de tela touch;

Não foi possível mostrar o funcionamento de todos os componentes exigidos para o projeto separadamente já que vários dos ítens de hardware foram encomendados pela internet, e nenhum dos mesmos havia chegado a tempo para os testes.Somente o teclado matricial de membrana, no qual foi comprado em uma loja física em Brasília.Assim para esse 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 -Outro 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. Desenvolvimento corrente

Os equipamentos e materiais que foram adquiridos para esse ponto de controle foram:

- Caixa de energia elétrica 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 6V e 12V).
- Fonte de 12V.

Por enquanto, apenas o teclado matricial foi integrado, parcialmente, ao sistema. Este ainda apresenta bugs no funcionamento, mas que pode ser corrigido com decorrer do tempo.

A tranca não foi implementada ainda, porém o acionamento (via software) está funcional e será explanado na próxima seção.

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.

Fig. 2. Raspberry Pi 3, modelo B.

IV. Software desenvolvido

- 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.

- Para o sistema do guarda-volumes, visando o uso do usuário, foi feito o programa 'terminal_tranca.c' que possibilita através de login na conta cadastrada previamente o acesso a abertura da porta do guarda-volumes (acionamento da tranca).

Para o correto funcionamento em sua execução, são necessárias as seguintes entradas: `./terminal_tranca -a <nome do usuario> <senha do usuario>`, devendo estar no diretorio do programa, via terminal no sistema Raspbian. O código do programa encontra-se no apêndice A.

- Visando implementação futura em servidor web, foi feito o programa 'sistema_contas.c' que possibilita o cadastro de novos usuários, e leitura de usuários já previamente cadastrados no sistema.

Para o correto funcionamento em sua execução, são necessárias as seguintes entradas para ler algum usuário já cadastrado: `./sistema_contas -l usuarios.txt <nome do usuario>`. Para cadastrar o usuario usa-se as entradas: `./sistema_contas -m usuarios.txt <nome do usuario> <senha> <numero de creditos> <senha do usuario>`, 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.

Os arquivos supracitados encontram-se em [3]

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/2_PCs/PC2>. Acesso em 1 de outubro de 2019.

APÊNDICE

Seguem os códigos citados na seção IV.

A. Programa ‘terminal_tranca.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 "gerencia_contas.h"
#define ARQUIVO_CONTAS     "usuarios.txt"
#define entrada_usuario argv[2]
#define entrada_senha argv[3]

int abreTranca(void);

int main(int argc, char * argv[]){
       const int creditos_minimos = 5;
       struct usuario usuario_corrente;

       if (argc < 3) {
              printf("Escolha uma opcao valida.\n");
              printf("Para utilizar o programa escreva no terminal:\n./terminal_tranca
-a <nome do usuario> <senha do usuario>\n");
              return -1;
       }
       if(argv[1][0]=='-'){
              switch(argv[1][1]){
                     case 'a':
                            usuario_corrente = leUsuario(ARQUIVO_CONTAS,entrada_usuario);
                            //printf("Usuario digitado: %s, senha digitada:%s\n",
entrada_usuario, entrada_senha);
                            //printf("ID: %d, usuario: %s, senha:%s, creditos: %d\n",
usuario_corrente.id, usuario_corrente.nome, usuario_corrente.senha,
usuario_corrente.creditos);
                            if (strcmp(entrada_usuario, usuario_corrente.nome) == 0){
                                   if (strcmp(entrada_senha, usuario_corrente.senha) ==
0){
                                          //FUNÇÃO PARA ABRIR A TRANCA AQUI!
                                          if (usuario_corrente.creditos >=
creditos_minimos){
                                                 printf("Abre-te Sesamo!\n");
```

```
                                        //abreTranca();
                                }
                                else{
                                        printf("Não é possivel utilizar o
sistema, usuario sem creditos o suficiente. (Pelo menos %d creditos)\n",
creditos_minimos);

                                }
                        }
                        else{
                                printf("Senha incorreta!\n");
                        }
                }
                else{
                        printf("Usuario nao encontrado.\n");
                        }
                break;
                defaut:
                        printf("Escolha uma opcao valida.\n");
            printf("Para utilizar o programa escreva no terminal:\n./terminal_tranca
-a <nome do usuario> <senha do usuario>\n");
            }
      }

      else {
            printf("Para utilizar o programa escreva no terminal:\n./terminal_tranca
-a <nome do usuario> <senha do usuario>\n");
      }
}
```

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

```
#include <stdio.h>
#include <stdlib.h>
#define entrada_arquivo argv[2]
#define entrada_usuario argv[3]
#define entrada_senha argv[4]
#define entrada_creditos argv[5]

#include "gerencia_contas.h"


int main(int argc, char * argv[]){
      struct usuario usuario_corrente;
      int sucesso;

      if (argc < 3) {
            printf("Escolha uma opcao valida.\n");
            printf("Para utilizar o programa escreva no terminal:\n- Para ler algum
usuario cadastrado: ./sistema_contas -l usuarios.txt <nome do usuario>\n- Para cadastrar
o usuario: ./sistema_contas -m usuarios.txt <nome do usuario> <senha>\n");
            return -1;
      }
      if(argv[1][0]=='-'){
            switch(argv[1][1]){
                  case 'l':
                        usuario_corrente =
leUsuario(entrada_arquivo,entrada_usuario);
                        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;
                           }
                           sucesso =
adicionaUsuario(entrada_arquivo,entrada_usuario,entrada_senha, atoi(entrada_creditos));
                           if (sucesso != 0) {printf("Erro no cadastro!\n");}
                           //printf("ID: %d\n", usuario_corrente.id);
                    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");
             }
       }
       else {
             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");
       }
}
```

C. 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 0;
       }
```

```
}
```