DRUG CARE

Controle de medicamentos

Paulo Henrique Moreira de de Carvalho Faria
Universidade de Brasília - FGA
oluap.ph@gmail.com

Rodrigo Bonfácio de Medeiros
Universidade de Brasília - FGA
rodrigo_medeiros92@hotmail.com

I. RESUMO

O presente ponto de controle descreve os avanços finais implementadas no projeto *Drugcare*, que consiste no gerenciamento inteligente de medicamentos, utilizando *Raspberry Pi [6]*. Nessa última etapa de desenvolvimento além da estrutura finalizada, o código de controle foi terminado e o *display* instalado para realizar a *interface*. Ainda nesse relatório final, foram relatados os resultados obtidos com os testes, dificuldade e limitações encontradas na construção do equipamento.

II. INTRODUÇÃO

Se tratando de um sistema que torna dinâmico a ingestão e armazenamento de medicamentos, o sistema *Drugcare* é um sistema que automatiza o modo de aquisição e estoque de medicamentos, bem como uma diretriz inicial para o protocolo de catálogo, controle de fluxo e horários de ingestão.

A automação para estoque de determinado artigo de utilidades é um processo que se torna vital na dinamização de sistemas de distribuição. Assim, todo o sistema empregado para esse processo deve ser projetado de forma sincronizada com o sistema de controle, seguindo um fluxograma de funcionamento, com o intuito de evitar atrasos de fluxo e falhas repentinas. [1]

No sistema *Drugcare* o medicamento vai estar sujeito a três processos: catálogo e rotinas, estoque e extração, como mostra a figura 1. O processo de catálogo e rotinas, recebe dados do usuário e armazena em arquivos do sistema para controle dos medicamentos em estoque e os coloca à disposição para criação de rotinas de ingestão. Adjacente a isso, o processo de extração utiliza um mecanismo controlado para disponibilizar os medicamentos em estoque de acordo com as rotinas em vigor.

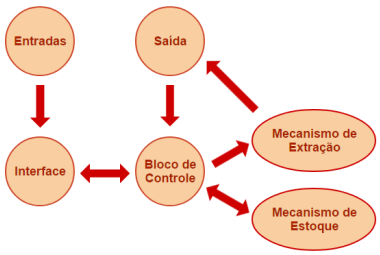


Figura 1 – Fluxograma mostrando o funcionamento do sistema Drugcare.

III. DESENVOLVIMENTO

Reforçando os blocos anteriormente citados, novos ajustes e modificações foram feitas para o resultado final, dentre eles podemos destacar: calibração dos motores de extração, encapsulamento externo e acabamento, sincronização do *display touch*, instalação de *LEDs* de sinalização e reformulação do código de controle.

1. Descrição do *hardware* e estrutura

1.1 Bloco de *interface*

O bloco de *interface*, o último a ser implementado, utilizou um *display touch capacitivo - 7 INCH HDMI LCD (B) [5]*,figura 2. O dispositivo apresentou um ótimo funcionamento e compatibilidade com a placa de controle. Apesar das vantagens, uma interface gráfica dedicada não foi finalizada a tempo para usufruir de toda a utilidade do display, além de um problema relacionado à demanda de corrente da fonte, que apresentou potência insuficiente a que constava nas especificações do fornecedor [5] que será melhor explicada na seção de resultados.

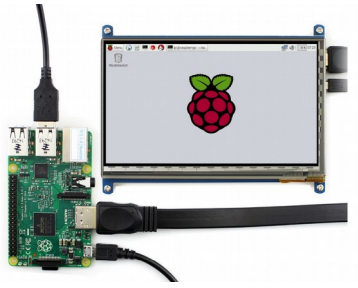


Figura 2 – Display touch capacitivo - 7 INCH HDMI LCD (B) conectado à raspberryPi.[5]

1.2 Bloco de controle

O bloco de controle é composto por um *Raspberry Pi 3 B+*, mostrado na figura 3, que é responsável por todo o controle e comunicação da máquina. Um *CI* com dois demultiplexadores (74LS139) [4], alimentação externa, foi usado para selecionar as saídas dos *LEDs* indicadores e o acionamento dos motores de extração, a fim de economizar portas GPIO,os mecanismos serão explicados mais adiante.

Figura 3 – Raspberry Pi 3 B+ usado no controle do Sistema. [6]

1.3 Bloco de extração

O bloco de extração é composto por um mecanismo de alavancas espelhadas que realizam o mesmo movimento quando solicitadas por apenas um ponto de tensão. Agora, o servo motor (TOWER PRO SG 90, figura 5) [3] possuirá um braço que ainda em uma mesma rotula irá impulsionar duas alavancas que obedecem a solicitação de torque de cada porta do alçapão. Todos os alçapões compartilham uma mesma rampa de saída que possui acesso manual para o usuário, como mostra a figura 3 e 4. O bloco é todo em madeira, exceto os motores, a figura 7 mostra o bloco final desenvolvido

Os quatros servos motores, com alimentação externa de 5V, serão controlados por dois pinos GPIO (4 e 17) de controle e um pino de PWM,pino 18 que é dedicado para isso. O demultiplexador fará a seleção do PWM de acordo com a lógica do pino para fazer o motor girar, como mostra a figura 6.

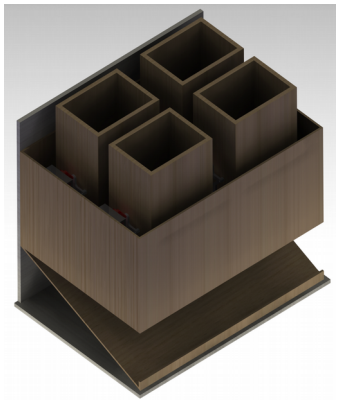


Figura 3 – Bloco de extração completo desenvolvido no software de CAD - CATIA.

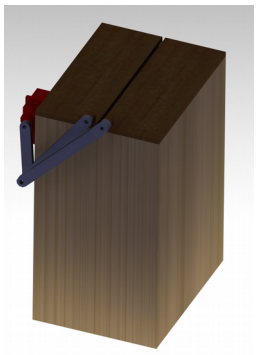
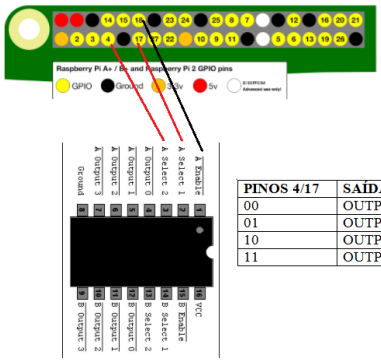


Figura 4 – Mecanismo de extração desenvolvido no software de CAD – CATIA.

Figura 5 – Servo motor – TOWER PRO SG 90 [3].



PINOS 4/17	SAÍDA PWM	MOTOR
00	OUTPUT 0	1
01	OUTPUT 1	2
10	OUTPUT 2	3
11	OUTPUT 3	4

Figura 6 – Esquemático simplificado com a tabela verdade para o controle dos motores.

Figura 7 – Bloco de extração completo com o display touch embutido.

1.4 Bloco de estoque

O bloco de estoque, composto por quatro compartimentos que tem dimensões que varrem todos os tamanhos de embalagens. Cada compartimento possui um par de *LED*s infravermelho, receptor e emissor que estabelecem um feixe entre eles, figura 8. Toda vez que o feixe é interrompido o sistema confere que o medicamento foi adicionado ou devolvido. Além disso cada compartimento também contém *LED*s verdes, anteriormente foi usado *RGB*, que orientam o usuário: aceso, indica o compartimento que está livre para armazenamento ou devolução.A alimentação dos *LED's* de orientação foi externa (5V) e o par emissor receptor foi interna(3.3V) para a entrada, nos pinos das *Raspberry* (na interrupção) também ser de 3.3V e não queimar as portas usadas (27,22,5,6).

O mesmo esquema de demultiplexação dos motores foram usados no acendimento dos *LED's* de indicação, com o uso das pinos do segundo multiplexador do CI 74LS139, os pinos GPIO de controle foi o 5 e 6

A tampa contém um orifício que possui vários tentáculos flexíveis, com o objetivo de guiar o usuário a inserir o medicamento de forma centralizada, figura 8.

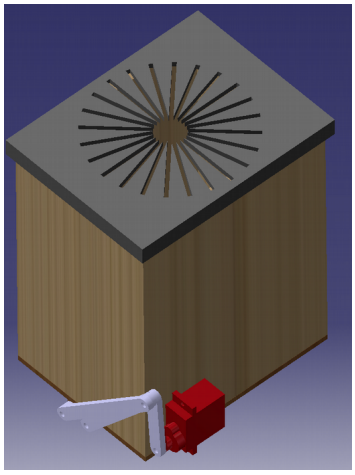


Figura 8 – Mecanismo guia com os tentáculos representados.

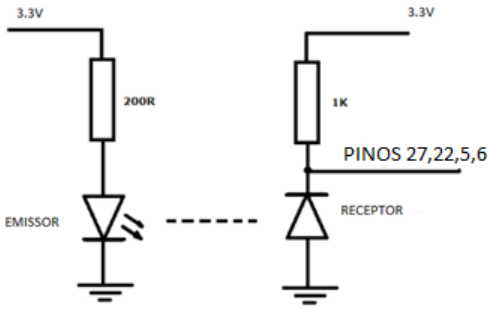


Figura 9 – Esquema simplificado do LED emissor e receptor para o controle do estoque.

2. Descrição do *Software*

O software foi desenvolvido no ambiente *LINUX*, com a distruibuição *RASPBIAN* [7] da *RaspberryPi*. A linguagem utilizada foi C, com auxílio da ferramento *GCC*.

A figura 9 mostra o fluxograma, completo do programa. Inicia com um menu de acesso do usuário que tem como opções cadastrar medicamento, com o nome; ano e mês de validade; número de comprimidos ; rotina (de quanto em quanto tempo o remédio é tomado. Após isso o mecanismo de estoque são ativados, ou seja, o LED do compartimento livre é aceso a espera da inserção de medicamento que é verificado quando há uma interrupção no feixe de infravermelho. As outras opções são a exibição e exclusão desses cadastro.

O programa está dentro de um loop infinito, uma *thread* concorrente ao programa foi criada para sempre atualizar a hora do sistema e verificar a rotina, ou seja, verificar se o remédio está vencido e avisar o usuário. A outra função da *thread* é verificar se é hora de tomar um remédio ativando o mecanismo de extração, ou seja, abre e fecha o compartimento com o servo motor, após isso os dados são atualizados, como o número de comprimidos e um relatório é criado. O código do programa se encontra em anexo.

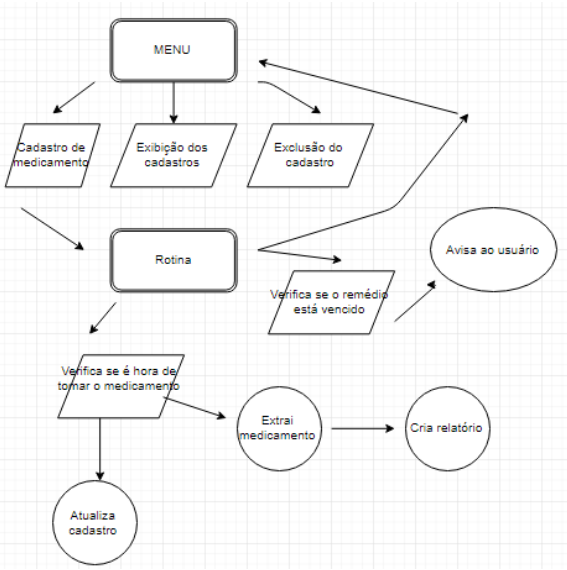


Figura 10 – Fluxograma completo do software

IV. RESULTADOS

Para a validação do projeto proposto, o sistema foi montado em partes. Primeiramente foi testado o *display* que funcionou perfeitamente, após isso o bloco de estoque foi montado e os resultados com vários testes foram satisfatórios.

Por último foi montado o bloco de extração (com testes anteriores satisfatórios), que teve resultados inesperados. Na ativação do PWM do motor a *Raspberry* ficava instável e desligava, uma possível causa desse problema seria alta demanda de corrente do display, no seu *datasheet* constava que uma fonte de 5V-2A era o suficiente para seu funcionamento e da *RaspberryPi*, foram testadas várias fontes

que forneciam 2A ou mais, mas mesmo assim o sistema não funcionou, talvez pela qualidade das fontes ou uma demanda de corrente mais alta da especificada no *Datasheet.*

V. CONCLUSÃO

Apesar do problema com *PWM* do motor, fazendo o sistema ficar instável, grande parte do projeto foi satisfatório. Os blocos separados funcionaram perfeitamente, e com o sistema totalmente montado apenas o bloco de extração teve problema.

Daremos continuidade no projeto, aperfeiçoando-o e corrigindo os problemas encontrados.

Em suma foi de grande aprendizado a montagem do sistema *Drugcare*, nos dando uma idéia importante dos sistemas embarcados e no uso da *RaspberryPi* e *Linux.*

REFERÊNCIAS

[1] HARADA, J.B: FERNANDA. SCHOR: PAULO. O Problema da autoadministração de medicamentosor idosos com baixa visão e cegueira sob a ótica do design centrado no humanoJ. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[2] Arduino and Processing: https://playground.arduino.cc/Interfacing/Processing Acesso ás 22 horas em 20 de Junho de 2017.

[3] Datasheet SG90 : http://akizukidenshi.com/download/ds/towerpro/SG90.pdf> Acesso ás 21:30 em 20 de Junho de 2017

[4] Datasheet 74LS139: : http://www.datasheetcatalog.com/info_redirect/datasheet/motorola/SN574LS139J.pdf.shtml> Acesso ás 21:00 em 20 de Junho de 2017

[5] Datasheet *7 INCH HDMI LCD (B)* http://www.waveshare.com/wiki/7inch_HDMI_LCD_(B)_(Firmware_Rev_2.1)_User_Manual Acesso ás 20:10 em 02 de Julho de 2017
.

[6] Datasheet RaspberryPi: <http://docs-europe.electrocomponents.com/webdocs/14ba/0900766b814ba5fd.pdf> Acesso ás 17:00 em 02 de Julho de 2017

[7] Raspbian https://www.raspbian.org/ Acesso ás 17:30 02 de Julho de 2017

VI. APÊNDICE

```
#include <stdio.h>     //
#include <stdlib.h>    //
#include <string.h>    //
#include <fcntl.h>     //   Bibliotecas padrões
#include <unistd.h>    //
#include <sys/types.h> //
#include <signal.h>    //
#include <time.h> // Biblioteca para a hora do sistema
#include <wiringPi.h> // Biblioteca para os pinos GPIO

#define PWM 18 // Pino GPIO do PWM (demux)
#define SEL_1_LED // Pino de seleção do LED (demux)
#define SEL_2_LED // Pino de seleção do LED (demux)
#define SEL_1_MOTOR // Pino de seleção do MOTOR (demux)
#define SEL_2_MOTOR // Pino de seleção do MOTOR (demux)
#define LED 27 // Pino para ligar ativar a saida dos LEDS(demux)
#define INT_1 22 // Pino GPIO da interrupção do recipiente 1
#define INT_2 5 // Pino GPIO da interrupção do recipiente 2
#define INT_3 6  // Pino GPIO da interrupção do recipiente 2
#define INT_4 13   // Pino GPIO da interrupção do recipiente 2

#define N_REC 4 // Definição do número de recipientes


int ocupado[N_REC]; // Varíavel para dizer um recipiente está ocupado


//---------Variável para a hora do sistema----------------

struct tm *local;
time_t t;
int hora;
int minuto;
int dia;
int mes;
int ano;

int hora_aux[N_REC];
int minuto_aux[N_REC];
int dia_aux[N_REC];
int mes_aux[N_REC];
int ano_aux[N_REC];

int hora_rotina[N_REC];
int mes_rotina[N_REC];
int ano_rotina[N_REC];
int quant_rotina[N_REC];


int k=1; // Variável para especificar qual recipiente inserir o remédio

// Declara a estrutura do cadastro do remédio
```

```
struct cadastro
{
      char remedio[30];
      int quant;
      int mes;
      int ano;
      int rotina;
};


struct cadastro recipiente[N_REC];// Declara o vetor para o número de
recipientes
int j; // Váriavel de controle para imprimir o cadastro na tela
int n_cadastro=0; // Varíavel de controle para cadastro


// Thread da bilbioteca WIringPI para atualizar a hora do sistema , verificar
se o remédio está vencido e acionar os motores
PI_THREAD (myThread)
{

      piLock (0) ; // MUtex da biblioteca WIringPI

      int a;                  //
      t=time(NULL);           //Atualiza a hora do sistema
      local=localtime(&t);    //

       hora2=local->tm_hour; //
       minuto2=local->tm_min; //
       dia2=local->tm_mday; //
       mes2=local->tm_mon; //  GUarda as varíaveis de tempo nas varaisveis do
programa
       mes2=mes2+1; //
       ano2=local->tm_year; //
       ano2=ano2+1900;  //

      for(a=0;a<N_REC;a++)
      {
            if (mes_rotina[a] == mes2 && ano_rotina[a] == ano2)
            {
            ven[a]=1;
            mes_rotina[a]=0;
            ano_rotina[a]=0;
            }
      }
      /* Rotina para o motor que não pode ser desenvolvido  por causa do
probleMAa do PWM ( A FUNÇÃO MOTOR PODE SER ENCONTRADA NO PONTO DE CONTROLE 3


      */
      piUnlock(0); // LIbera o MUtex da biblioteca WIringPI
}

// FUnção para verificar interrupção do feixe de infravermelho
```

```
void inter()
{
      system("clear");
      printf("\nInsira o remédio no local indicado\n");
      if (n_cadastro == 0)
        {
         digitalWrite(LED,LOW);
      digitalWrite(SEL_1_LED,LOW);
      digitalWrite(SEL_2_LED,LOW);
      while(1){

            if(digitalRead(INT_1) == 1)
            {
            system("clear");
            digitalWrite(LED,HIGH);
            printf("*Remédio inserido*\n\n");
            sleep(3);
            return;
            }
         }
      }
      else if (n_cadastro == 1)
        {
         digitalWrite(LED,LOW);
      digitalWrite(SEL_1_LED,LOW);
      digitalWrite(SEL_2_LED,HIGH);
      while(1){

            if(digitalRead(INT_2) == 1)
            {
            system("clear");
            digitalWrite(LED,HIGH);
            printf("*Remédio inserido*\n\n");
            sleep(3);
            return;

            }
             }
      }

      else if (n_cadastro == 2)
        {
         digitalWrite(LED,LOW);
      digitalWrite(SEL_1_LED,LOW);
      digitalWrite(SEL_2_LED,HIGH);
      while(1){

            if(digitalRead(INT_3) == 1)
                  {
            system("clear");
            digitalWrite(LED,HIGH);
            printf("*Remédio inserido*\n\n");
            sleep(3);
```

```
            return;
                  }
            }
       }
      else if (n_cadastro == 4)
        {
         digitalWrite(LED,LOW);
      digitalWrite(SEL_1_LED,HIGH);
      digitalWrite(SEL_2_LED,HIGH);
      while(1){

            if(digitalRead(INT_4) == 1)
            {
            system("clear");
            digitalWrite(LED,HIGH);
            printf("*Remédio inserido*\n\n");
            sleep(3);
            return;
            }
           }
      }

}

// Função para cadastro do remédio no vetor recipiente
void cadastra_remedio(struct cadastro rec)
{

      FILE *fp;
      char buffer[100] = " ";

      hora_aux[n_cadastro]=hora;
      minuto_aux[n_cadastro]=minuto;
      dia_aux[n_cadastro]=dia;
      mes_aux[n_cadastro]=mes;
      ano_aux[n_cadastro]=ano;

  ocupado[n_cadastro]=1;
      recipiente[n_cadastro]=rec;
      mes_rotina[n_cadastro]=recipiente[n_cadastro].mes;
      ano_rotina[n_cadastro]=recipiente[n_cadastro].ano;
      quant_rotina[n_cadastro]=recipiente[n_cadastro].quant;

      sprintf(buffer,"Cadastro_%d.txt",n_cadastro+1);
      fp=fopen(buffer,"w");
      fprintf(fp,"Nome do remédio: %s\n",recipiente[n_cadastro].remedio);
      fprintf(fp,"Data de validade: %02d/
%04d\n",recipiente[n_cadastro].mes,recipiente[n_cadastro].ano);
      fprintf(fp,"Quantidade de comprimidos:
%02d\n",recipiente[n_cadastro].quant);
      fprintf(fp,"Rotina: de %02d em %02d
horas\n",recipiente[n_cadastro].rotina,recipiente[n_cadastro].rotina);
```

```
      fprintf(fp,"Dia e horário inseridos: %02d/%02d/%04d as %02d horas e %02d
minutos",dia_aux[n_cadastro],mes_aux[n_cadastro],ano_aux[n_cadastro],hora_aux[n
_cadastro],minuto_aux[n_cadastro]);

      fclose(fp);
}

// FUnção para ler os dados do novo remédio
// Retorna um remédio com os dados preenchidos

struct cadastro le_cadastro()
{
      struct cadastro auxiliar;

      t=time(NULL);
      local=localtime(&t);
      hora=local->tm_hour;
      minuto=local->tm_min;
      dia=local->tm_mday;
      mes=local->tm_mon;
      mes=mes+1;
      ano=local->tm_year;
      ano=ano+1900;

      inter();

       printf("\n\n");
       printf("Nome do remédio: ");
       fflush(stdin);
       scanf("%s",auxiliar.remedio);
       printf("Mês de validade(mm) - Exemplo (02): ");
       fflush(stdin);
       scanf("%d",&auxiliar.mes);
       printf("Ano de validade(yyyy): Exemplo (2020): ");
       fflush(stdin);
       scanf("%d",&auxiliar.ano);
       printf("Número de comprimidos: ");
       fflush(stdin);
       scanf("%d",&auxiliar.quant);
       printf("Criar rotina (em Horas): ");
       fflush(stdin);
       scanf("%d",&auxiliar.rotina);
       system("clear");
       hora=local->tm_hour;
       minuto=local->tm_min;
       dia=local->tm_mday;
       mes=local->tm_mon;
       mes=mes+1;
       ano=local->tm_year;
       ano=ano+1900;
       return auxiliar;
}

// Função   para imprimir todos os cadstros do remédio na tela
```

```
void imprime_cadastro() {

                        printf("\n\n");
                        printf("===Recipiente 1=== \n");
                        system("cat Cadastro_1.txt");
                        sleep(3);
                        printf("\n\n");
                        printf("===Recipiente 2=== \n");
                        system("cat Cadastro_2.txt");
                        printf("\n\n");
                        sleep(3);
                        system("clear");
                        printf("===Recipiente 3=== \n");
                        system("cat Cadastro_3.txt");
                        printf("\n\n");
                        sleep(3);
                        printf("===Recipiente 4=== \n");
                        system("cat Cadastro_4.txt");
                        printf("\n\n");
                        sleep(3);
                        system("clear");

}
// Função para exibir o menu de opções
int menu() {
                int opcao;

                printf("\n\n ==== MENU DE OPCOES ====\n");
                printf("1 - Cadastrar medicamento \n");
                        printf("2 - Exibir cadastros \n");
                printf("3 - Excluir cadastro \n");
                printf("\nO que deseja fazer? ");
                scanf("%d", &opcao);

                return opcao;
            }

// Função para remover um cadastro do vetor de posição
            void remover(int posicao) {


                if ((posicao < 1) || (posicao > N_REC)) {
                    printf("\nRecipiente inválido\n");
                                sleep(3);
                                system("clear");
                    return;
                }

                if (posicao == N_REC) {
                                system("rm Cadastro_4.txt");
                                ocupado[N_REC-1]=0;
                                printf("\nRecipiente 1 Liberado - Retire o
medicamento,caso houver\n");
                                sleep(3);
```

```
                                    system("clear");
                                    return;
                }

                        else if (posicao == N_REC-1){
                                    system("rm Cadastro_3.txt");
                                    ocupado[N_REC-2]=0;
                                    printf("\nRecipiente 2 Liberado - Retire o
medicamento,caso houver\n");
                                    sleep(3);
                                    system("clear");
                                    return;
                         }

                         else if (posicao == N_REC-2){
                                    system("rm Cadastro_2.txt");
                                    ocupado[N_REC-3]=0;
                                    printf("\nRecipiente 3 Liberado - Retire o
medicamento,caso houver\n");
                                    sleep(3);
                                    system("clear");
                                    return;
                        }
                        else if (posicao == N_REC-3){
                                    system("rm Cadastro_1.txt");
                                    ocupado[N_REC-4]=0;
                                    printf("\nRecipiente 4 Liberado - Retire o
medicamento,caso houver\n");
                                    sleep(3);
                                    system("clear");
                                    return;

                        }

}


//Sinal para encerrar o programa
void encerrar(){
      puts("\nFechando programa...\n");
      sleep(2);
      system("clear");
      exit(0);
}

int main()
{


  unsigned int choice; // Variável de controle para escolha do parâmetro
  struct cadastro novo_cadastro;
  int opcao_selecionada;
  int rem = 0;
```

```
  int r;
  int aux=0;

  if (wiringPiSetupGpio() == -1) // INicializa a biblioteca WIringPi
   exit (1) ;
  pinMode (SEL_1_LED,OUTPUT); // SEL_1 como saída para o demux
  pinMode (SEL_2_LED,OUTPUT); // SEL_2 como saída para o demux
  pinMode (LED,OUTPUT);     PWM_1 como saída para o demux
  pinMode (INT_1,INPUT); // pino para verificar a interrupção como entrada na
rasp
  pinMode (INT_2,INPUT); // pino para verificar a interrupção como entrada na
rasp
  pinMode (INT_3,INPUT); // pino para verificar a interrupção como entrada na
rasp
  pinMode (INT_4,INPUT); // pino para verificar a interrupção como entrada na
rasp
  digitalWrite(SEL_1_LED,LOW);
  digitalWrite(SEL_2_LED,LOW);
  digitalWrite(LED,HIGH);

  pid_t pid[N_REC]; // Variável para a criação de processos

  signal(SIGINT,encerrar); // Sinal para encerrar o programa
     system("clear");

//----------------------Menu do sistema---------LOOP
INFINITO-----------------------------------------------

do {
                opcao_selecionada = menu();
                switch(opcao_selecionada) {

                                        system("clear");
                        case 0: break;

                        case 1: // código para inserir
                                  system("clear");
                                  aux=0;
                                  n_cadastro=0;
                                    while(ocupado[n_cadastro]==1)
                                    {
                                       n_cadastro++;
                                       if(n_cadastro == 4){
                                          printf("\n NÃO FOI POSSÍVEL
COMPLETAR O CADASTRO - RECIPIENTES CHEIOS \n");
                                                sleep(2);
                                                system("clear");
                                                aux=1;
                                           }
                                    }
                                       if(aux==1)
                                       break;
                                       else
                                       {
```

```
                                        novo_cadastro = le_cadastro();
                                        cadastra_remedio(novo_cadastro);

      hora_rotina[n_cadastro]=novo_cadastro.rotina;

      mes_rotina[n_cadastro]=novo_cadastro.mes;

      ano_rotina[n_cadastro]=novo_cadastro.ano;

      quant_rotina[n_cadastro]=novo_cadastro.quant;
                                        break;
                                    }
                            case 2: // código para exibir
                                        system("clear");
                                        imprime_cadastro();
                                        break;
                            case 3: // código para excluir
                                        system("clear");
                                        printf("Digite o recipiente que deseja
excluir o cadastro (1-4): ");
                                        scanf("%d", &r);
                                        system("clear");
                                        remover(r);
                                        break;
                            default:
                                        printf("----- OPCAO INVALIDA -----");
                                        system("clear");
                }
      }
      while (opcao_selecionada != 0);

  return 0;
}
```