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 e soluções implementadas no projeto Drugcare, que consiste no gerenciamento inteligente de medicamentos, utilizando Raspberry Pi. Nessa terceira etapa de desenvolvimento a estrutura teve sua construção iniciada e o mecanismo de extração uma modificação, além de um avanço considerável no controle dos motores, sensores de armazenamento e controle.

II. INTRODUÇÃO

Se tratando de um sistema que torna dinâmico, a ingestão e armazenamento de medicamentos, o sistema Drugcare demandou a criação de um sistema que automatizasse 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 maquinário 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. 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.

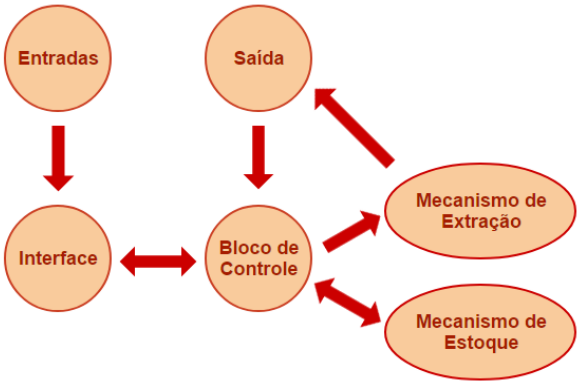

Figura 1 – Fluxograma Drugcare

III. DESENVOLVIMENTO

Para o desenvolvimento inicial do bloco de controle, que a princípio simula a interface de catálogo e rotina, foi necessária a utilização de ideias de escrita em arquivo e conexão GPIO para formar o protocolo de acesso do usuário a máquina. Quanto ao mecanismo de extração, foi necessária a idealização de um sistema de alçapões, uma para cada um dos quatro compartimentos de estoque do protótipo. Cada alçapão é independente e controlado por um servo-motor que é acionado quando solicitado um medicamento.

1. Descrição do hardware e estrutura

1.1 Bloco de interface

A interface também sofreu uma mudança, anteriormente feita por um smartphone, agora será protagonizada por um display touch compatível com o Raspberry Pi que exibira uma interface gráfica dedicada.

Figura 2 – Display touch.

1.2 Bloco de controle

O bloco de controle é composto por um Raspberry Pi 3 B+, que é responsável por todo o controle e comunicação da máquina. Os outros componentes citados em relatórios passados foram considerados desnecessários e retirados do escopo.

Figura 3 – Raspberry Pi 3 B+

1.3 Bloco de extração

O bloco de extração sofreu uma modificação considerável, que antes era composto por um mecanismo de alavancas espelhadas que realizam o mesmo movimento quando solicitadas por apenas um ponto de tensão. Agora, o servo-motor, figura 6, 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. O bloco é todo em madeira, exceto os motores.

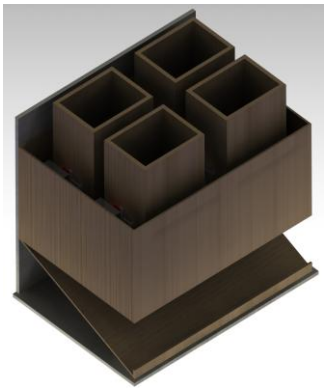

Figura 4 – Bloco de extração completo.

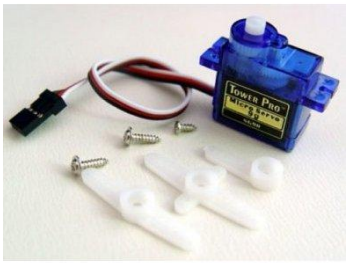

Figura 6 – Servo-motor SG - 9.

1.4 Bloco de Estoque

O bloco de estoque é composto por quatro compartimentos que tem dimensões que varrem todos os tamanhos de embalagens. Cada compartimento possui um LED de infravermelho, e transistor receptor que estabelecem um feixe entre eles, figura 7. Toda vez que o feixe é interrompido o sistema confere que o medicamento foi adicionado ou devolvido. Além disso cada compartimento também contém LEDs RGB indicativos, que dependendo da cor orientam o usuário: amarelo, para aviso de devolução e inserção; verde, para indicar o compartimento que está escalado na rotina no alarme de ingestão; vermelho, indica o compartimento que já foi ingerido. Este bloco ainda está em construção.

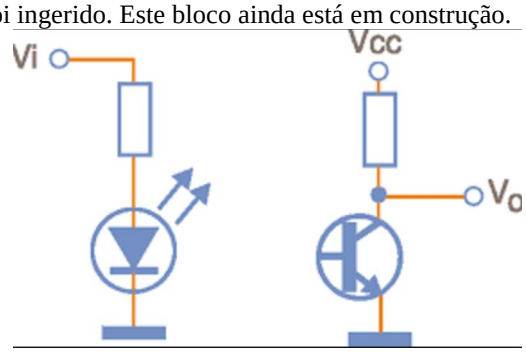

Figura 7 – Servo-motor SG - 9.

2. Descrição do Software

Uma segunda versão de teste (v 2.0) - com a simulação de um motor - do software foi escrito em linguagem C para Linux e compilado com o auxílio da ferramenta de desenvolvimento GCC. A figura 8 mostra o fluxograma do programa. As modificações feitas em relação a versão 1.0 (que se encontra no ponto de controle 2) é o uso do PWM com auxílio da biblioteca "wiringPi" para controle do servo-motor, no caso quando o duty-cycle é de 2ms numa freqüência de 50 Hz o motor gira horário, quando é 1ms o motor gira anti-horário. A inclusão de uma interrupção com um esquema de emissor e receptor de infravermelho, para a indicação da entrada do remédio no recipiente, ou seja , o feixe infravermelho sempre deixa Vo em "alto" (3.3 V) quando há uma interrupção Vo vai para "baixo". O grande feito nesse ponto de controle foi o teste da interrupção e o controle do servo-motor, o código se encontra no final desse arquivo. A intenção para os próximos testes é modificar o software para os quatro recipientes com uso de *threads* , fazer testes de validação para diversos tipos de situações, otimizar a função de PWM e completar a interface gráfica com o usuário

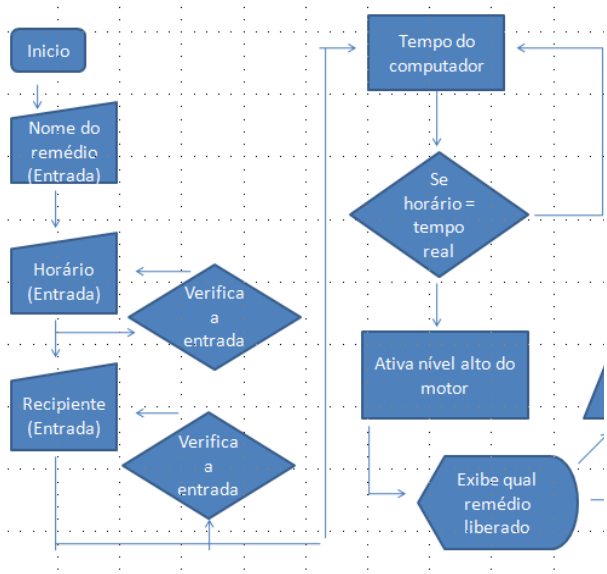

Figura 8 – Fluxograma V2.0

IV. RESULTADOS

Para a validação do projeto proposto, foi montado um circuito usando um servo-motor e o circuito emissor-receptor para a validação do software, os testes foram satisfatórios, pois a interrupção foi detectada e o motor foi controlado, porém o PWM deve ser otimizado para um controle maior, pois o softaware proposto só fornece dois movimentos de 90º horário anti-horário.

V. CONCLUSÃO

No presente ponto de controle, algumas mudanças importantes foram definidas, como a hardware interface com o usuário, com o uso de monitor *touch* . O esquema de extração e estoque ficam bem definidos.

O grande desafio são os casos de teste das situações que podem ocorrer; interface gráfica para o uso mais viável ao usuário e a simulação com diversos motores, ou seja, é necessário uma melhor otimização para software.

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.

.

VI. APÊNDICE

```
#include<stdio.h>     //Biblioteca padrão de entrada /saída
#include<time.h>      //Biblioteca para os uso do tempo do
          sistema
#include<stdlib.h>    //Biblioteca de propósito geral
#include<wiringPi.h>  //Biblioteca para os pinos GPIO
#include<string.h>    // Biblioteca para manipulação de strings

#define INT 7  // Define GPIO 4 para a interrupção
#define MOTOR 0 // Define GPIO 18 para o motor
```

```
// Função para a interrupção
int inter()
{
      printf("Insira o remÃ©dio no local indicado\n");

      while(1){
            if(digitalRead(INT) == 0)
            {
            printf("*RemÃ©dio inserido*\n\n");
            return 1;
            }
      }
}

// Função para o servo motor com PWM
int motor()
{
      int i;
      // PWM para o giro do motor horário ( abrir)

      for(i=0;i<100;i++)
      {
      digitalWrite( MOTOR, HIGH );
         usleep(2000);
         digitalWrite(MOTOR,  LOW );
      usleep(18000);
      }
      // PWM para o giro do motor anti-horário (fechar)
      for(i=0;i<100;i++)
      {
      digitalWrite( MOTOR, HIGH );
         usleep(1000);
         digitalWrite(MOTOR,  LOW );
      usleep(19000);
      }
      printf("*RemÃ©dio liberado*\n\n");
      return 1;

}

int main(void)
{
struct tm *local; // Struct para determinar a hora real do
            computador

int hora;       // Variável para a hora real do computador
int minutos;    // Variável para o minuto real do computador
int h_des=0;     // Variável para a hora desejada de tomar o
        remédio
int m_des=0;    // Variável para o minuto desejado de tomar
        remédio
char nome[30];  // Variável para o nome do remédio
```

```
int rec;         // Variável para dado recipente

FILE *fp;                       // Abertura do arquivo de relatório
fp=fopen("Relatorio.txt","w"); //

wiringPiSetup () ;  // Inicialização da biblioteca de GPIO

pinMode (INT, INPUT) ; // Define o pino INT como entrada
pinMode (MOTOR,OUTPUT); // Define o pino MOTOR como saída


time_t t;             //
t=time(NULL);         // Definindo o tempo real do computador
local=localtime(&t); //
inter(); // Chama a função de interrupção esperando ser inserido o remédio

printf("Digite o nome do remedio: "); // Entrada do nome do
                                      remédio
scanf("%s",nome);                          //

// Entrada da hora de tomar o remédio com a condição de hora
// inválida
do
{
  printf("Digite a hora desejada de tomar o remÃ©dio (00-23):");
  scanf("%d",&h_des);
  if (h_des > 23 || h_des< 0)
    printf("*Hora InvÃ¡lida*\n\n");
} while (h_des > 23 || h_des< 0);

printf("*Hora inserida*\n\n");

// Entrada do minuto de tomar o remédio com a condição de minuto
// inválida
do
{
  printf("Digite o minuto desejado(1-59):");
  scanf("%d",&m_des);
  if (m_des<1 || m_des>59)
    printf("*Minuto invalido*\n\n");
}while (m_des<1 || m_des>59);

printf("*Minuto inserido*\n\n");


hora=local->tm_hour; // Determinando a hora real
minutos=local->tm_min; // Determinado o minuto real

// Verifica se o horário de tomar é igual ao tempo real
if(hora == h_des && minutos == m_des)
{
    motor();   // Abre o motor liberando o remédio
    fprintf(fp,"Remedio tomado: %s\n",nome);
```

```
    fprintf(fp,"Horario tomado: %d:%d\n",h_des,m_des);
}


fclose(fp); // Fecha o arquivo

return 0;
}
```