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 quarta etapa de desenvolvimento a estrutura teve sua construção finalizada e o mecanismo de extração foi modificado como esperado. O controle dos motores foi consolidado e um mecanismo de guia para o sensor de armazenamento foi adicionado ao conjunto.

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,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.

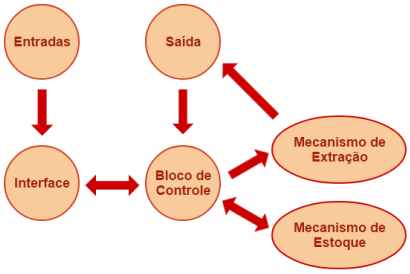


Figura 1 – Fluxograma Drugcare

III. DESENVOLVIMENTO

Para o desenvolvimento inicial do bloco de controle, que a princípio simula a interface de catalogo 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 exibirá uma interface gráfica dedicada, feita pela *processing*, uma linguagem de programação de código aberto utilizada para escrever programas utilizando representação gráfica.[2]

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 (TOWER PRO SG 90, figura 6)[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. O bloco é todo em madeira, exceto os motores.

Os quatros servos motores serão controlados por dois pinos GPIO (4 e 17) de controle e um pino de PWM,pino 18 que é dedicado para isso.Um demux, 74LS139 [4], fará a seleção do PWM de acordo com a lógica do pino para fazer o motor girar, como mostra a figura 7.

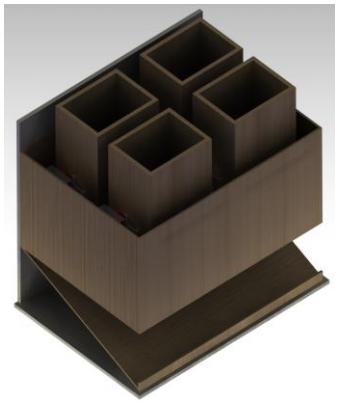


Figura 4 – Bloco de extração completo.

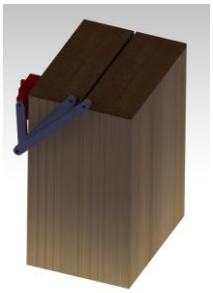
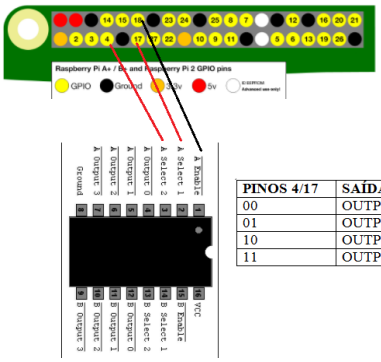


Figura 5 – Mecanismo de extração.

Figura 6 – Servo motor



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

Figura 7 – Esquemático simplificado com a tabela verdade para o controle dos motores.

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 par de LEDs infravermelho, receptor e emissor que estabelecem um feixe entre eles, como mostra a figura 8. Toda vez que o feixe é interrompido o sistema confere pelos pinos 19 e 26 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. A tampa contém um orifício que possui vários tentáculos flexíveis, com o objetivo de guiar o usuário a inserir o medicamento de forma centralizada.

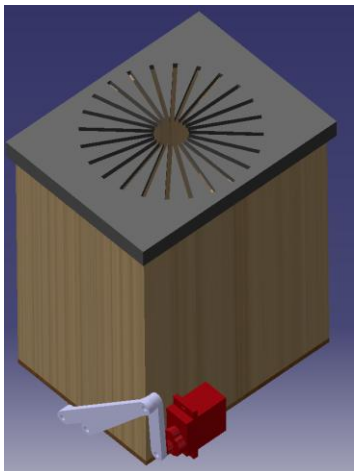


Figura 7 – Mecanismo guia.

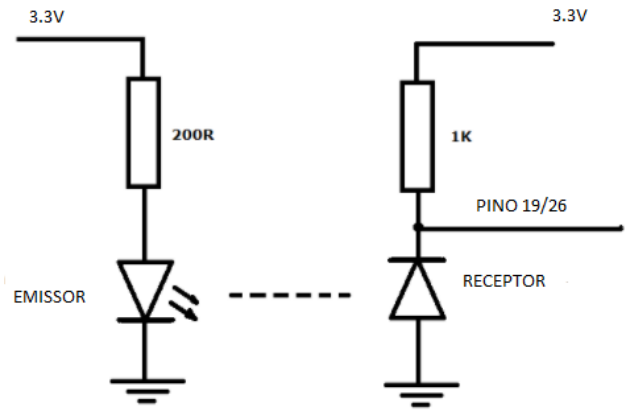


Figura 8 – Esquema simplificado do LED emissor e receptor para o controle do estoque.

2. Descrição do Software

A figura 8 mostra o fluxograma,ainda em desenvolvimento do programa. O Bloco "início" indica o início do processo, após isso é requerida as entradas do usuário (Nome do remédio, horário e o recipiente a ser guardado). Os losangos representam uma condição, se a entrada for inválida, como recipiente ou horário não existente, ele retorna a pedir a entrada ao usuário até a entrada ser válida.

Após os dados serem guardados, o código verifica se o horário indicado pelo usuário é igual ao tempo real do computador, ou seja, se é a hora do paciente tomar o remédio, caso a condição seja falsa, o programa verifica novamente até a condição ser verdadeira. Quando a condição for verdadeira é ativo um nível lógico alto que aciona o motor do recipiente para liberar o remédio, é impresso na tela qual remédio a ser tomado e um relatório é criado com dos dados do processo (nome do remédio e horário). No apêndice encontra a função para o controle dos motores e do bloco de estoque, ainda em desenvolvimento.

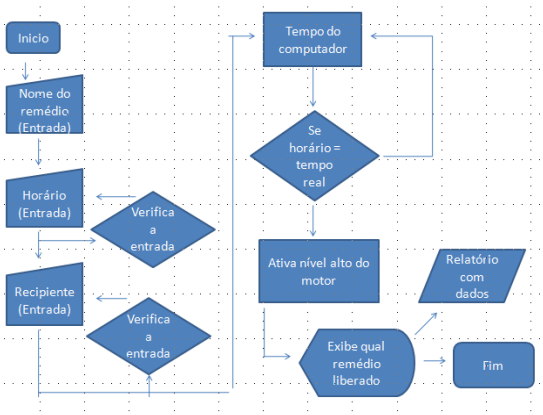


Figura 8 – Fluxograma do software

IV. RESULTADOS

Para a validação do projeto proposto, foi montado um circuito com a ativação de dois motores e do LED emissor/receptor já instalados na estrutura, todo o processo foi satisfatório, pois o controle está bem estabelecido, com o PWM do servo motor estável e a interrupção funcionando perfeitamente.

V. CONCLUSÃO

O projeto está com o desevolvimento satisfatório, para completá-lo falta terminar a estrutura e instalação dos últimos três emissores e recptores, otimizar e completar o código da descrição do software, observando os testes de caso.

Outro módulo importante, também em desenvolvimento, é a interface gráfica que será desenvolvido em conjunto com código para uma melhor performance.

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/SN54LS139J.pdf.shtml> Acesso ás 21:00 em 20 de Junho de 2017

.

VI. APÊNDICE

```
#include <wiringPi.h> // Biblioteca de pinos GPIO
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

#define PWM 18 // Define o pino 18 como PWM
#define SEL_1 4 // Define o pino 4 como seleção do demux
#define SEL_2 17 // Define o pino 17 como seleção do demux
#define REC_1 19 // Define o pino 19 como interrupção DO Recipiente 1


// Função para a interrupção
int inter(int rec)
{

      printf("Insira o remédio no local indicado\n");

  // Reméio so é detectado quando o feixe de emissor e receptor for
interceptado
      while(1){
            if(digitalRead(INT) == 1) // COndição quando ocorre a interceptação
            {
            printf("*Remédio inserido*\n\n");
            return 1;
            }
      }
}

int main (void)
{
  int rec;
  // Inicializa o sistema wiringPi para usar o pinos de GPIO
  if (wiringPiSetupGpio() == -1)
   exit (1) ;
  pinMode (SEL_1,OUTPUT); // Define o pino 4 como saída
  pinMode (SEL_2,OUTPUT); // Define o pino 17 como saída
  pinMode (REC_1,INPUT); // Define o pino 19 como entrada

  // seta o modo do pino para INPUT, OUTPUT ou PWM_OUTPUT
  pinMode(PWM,PWM_OUTPUT);
  // Configura o PWM para o modo mark:space
  pwmSetMode(PWM_MODE_MS);
  // seta o divisor para o clock do PWM
  pwmSetClock(4000);
  // seta o registrador de intervalo no gerador de PWM de 50 Hz (Frequência do
servo motor)
  pwmSetRange (96);
  // seta o contador de intervalo no gerador de PWM de 50 Hz
while(1)
```

```
{
inter(); // chama a função de controle do estoque
printf("Recipiente : \n");
scanf("%d",&rec); // Define qual o recipiente será aberto

//LIGA O MOTOR 1
if (rec == 1)
{
// LOGICA REC 1
digitalWrite(SEL_1,LOW); // PINO DE CONTROLE 4 DO DEMUX EM NíVEL BAIXO
digitalWrite(SEL_2,LOW); // PINO DE CONTROLE 17 DO DEMUX EM NíVEL BAIXO
    delay(1000);
     pwmWrite (PWM,20); // FREQUENCIA DO PWM ALTERADA PARA RODAR O SERVO E
ABRIR O COMPATIMENTO
    delay(1000);
     pwmWrite (PWM,9); // FREQUENCIA DO PWM ALTERADA PARA RODAR O SERVO E
FECHAR O COMPATIMENTO
    delay(1000);
     pwmWrite (PWM,0); // DESLIGAR O PWM
}

//LIGA O MOTOR 2
if (rec == 2)
{
// LOGICA REC 2
digitalWrite(SEL_1,LOW); // PINO DE CONTROLE 4 DO DEMUX EM NíVEL BAIXO
digitalWrite(SEL_2,HIGH); // PINO DE CONTROLE 17 DO DEMUX EM NíVEL ALTO
    delay(1000);
     pwmWrite (PWM,20); // FREQUENCIA DO PWM ALTERADA PARA RODAR O SERVO E
ABRIR O COMPATIMENTO
    delay(1000);
     pwmWrite (PWM,9); // FREQUENCIA DO PWM ALTERADA PARA RODAR O SERVO E
FECHAR O COMPATIMENTO
    delay(1000);
     pwmWrite (PWM,0); // DESLIGAR O PWM
}

//LIGA O MOTOR 3
if (rec == 3)
{
// LOGICA REC 3
digitalWrite(SEL_1,HIGH); // PINO DE CONTROLE 4 DO DEMUX EM NíVEL ALTO
digitalWrite(SEL_2,LOW); // PINO DE CONTROLE 17 DO DEMUX EM NíVEL BAIXO
    delay(1000);
     pwmWrite (PWM,20); // FREQUENCIA DO PWM ALTERADA PARA RODAR O SERVO E
ABRIR O COMPATIMENTO
    delay(1000);
     pwmWrite (PWM,9); // FREQUENCIA DO PWM ALTERADA PARA RODAR O SERVO E
FECHAR O COMPATIMENTO
    delay(1000);
     pwmWrite (PWM,0); // DESLIGAR O PWM
}
//LIGA O MOTOR 4
```

```
if (rec == 4)
{
// LOGICA REC 4
digitalWrite(SEL_1,LOW); // PINO DE CONTROLE 4 DO DEMUX EM NíVEL ALTO
digitalWrite(SEL_2,HIGH); // PINO DE CONTROLE 17 DO DEMUX EM NíVEL ALTO
    delay(1000);
     pwmWrite (PWM,20); // FREQUENCIA DO PWM ALTERADA PARA RODAR O SERVO E
ABRIR O COMPATIMENTO
    delay(1000);
     pwmWrite (PWM,9); // FREQUENCIA DO PWM ALTERADA PARA RODAR O SERVO E
FECHAR O COMPATIMENTO
    delay(1000);
     pwmWrite (PWM,0); // DESLIGAR O PWM
}
}
  return 0;
}
// LOGICA REC 4
digitalWrite(SEL_1,LOW); // PINO DE CONTROLE 4 DO DEMUX EM NÃ□VEL ALTO
digitalWrite(SEL_2,HIGH); // PINO DE CONTROLE 17 DO DEMUX EM NÃ□VEL ALTO
    delay(1000);
     pwmWrite (PWM,20); // FREQUENCIA DO PWM ALTERADA PARA RODAR O SERVO E
ABRIR O COMPATIMENTO
    delay(1000);
     pwmWrite (PWM,9); // FREQUENCIA DO PWM ALTERADA PARA RODAR O SERVO E
FECHAR O COMPATIMENTO
    delay(1000);
     pwmWrite (PWM,0); // DESLIGAR O PWM
}
}
  return 0;
}
```