MÁQUINA DE REFRIGERANTE COM TRAVA DE SEGURANÇA

Kewin Kuster, Rodrigo Sousa Santos

Programa de Graduação em Engenharia Eletrônica, Faculdade Gama
Universidade de Brasília
Gama, DF, Brasil
Email: kevinkiister0@gmail.com, rodrigo.sousa2711@gmail.com

1. JUSTIFICATIVA

A motivação inicial para a criação de uma máquina de refrigerantes com um sistema de controle de usuário partiu do reconhecimento do problema enfrentado por empresas alimentícias em locais públicos aglomerados como shoppings, hipermercados, grandes centros de comercio entre outros exemplos. O intuito da construção do sistema consiste em monitorar e controlar o consumo de bebidas em ambientes abertos, de modo a evitar que pessoas que não sejam clientes utilizem dos produtos disponibilizados como refil fornecidos pela loja. Uma vez que a utilização da máquina com a trava tornaria muito mais raros eventos como esses.

O sistema funciona através de uma simples implementação aos modelos de máquinas utilizadas hoje em dia, com a adição de uma câmera fotográfica capaz de realizar o processamento de um código adicionado aos copos fornecido pela empresa distribuidora da bebida, podemos realizar todo o controle de acesso a máquina de refrigerantes.

2. OBJETIVOS

O objetivo deste projeto é construir um sistema capaz de controlar uma máquina de refrigerante de refil na forma de evitar que pessoas burlem o sistema, realizando desse modo um maior controle sobre o fluxo de serviço prestado pela máquina.

3. REQUISITOS

Para realização do projeto, ao se realizar a compra do copo para refil, será criado um qr code e nele será inserido o horário da compra e o horário limite de uso da máquina, sendo isto uma validação que irá durar um certo período de tempo sem prejudicar o modo refil de utilização. Este qr code será impresso e colado no copo no momento da compra. Após o cadastramento do qr code, a pessoa passará o copo em um leitor de qr code, onde será utilizada uma câmera para leitura. Esta câmera estará conectada a uma raspberry pi3 para o processamento da imagem e análise dos dados. A mesma será utilizada para acionar a máquina de refrigerante, ativando um módulo relé conectada a uma bom liberando o funcionamento.

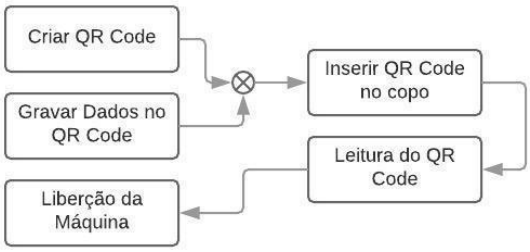

Fig. 1. Fluxograma do projeto.

4. BENEFÍCIOS

O intuito da realização deste projeto está em conseguir que a utilização de um produto seja dada somente pelos clientes da loja de alimentos, fazendo com que haja a diminuição ou até mesmo a extinção desse mesmo consumo por terceiros, para que não haja prejuízo para a empresa que optar pela utilização do mesmo.

5. ASPECTOS DE HARDWARE

Segue abaixo uma tabela de materiais utilizados na implementação do projeto, quantidades e uma breve descrição de cada componente.

Tabela 1. Materiais utilizados no projeto.

Componentes	Descrição
RaspberryPi3	Sistema Embarcado
Câmera	Leitura do QrCode
Módulo Relé	Acionamento da Bomba
Bomba de Aquário	Líquido liberado
2 Leds	Ligado ou Desligado
Botão	Acionamento da Bomba

5.1. Raspberry Pi3

Raspberry Pi é um computador de baixo custo e que tem o tamanho de um cartão de crédito desenvolvido no Reino Unido pela Fundação Raspberry Pi. Ela possui Wifi e bluetooth integrado, um processador quad-core de 64 bits (Broadcom BCM2837), clock de 1.2 GHz e ainda conta ainda com uma arquitetura avançada, da Cortex-A53. Na parte gráfica, usa um processador gráfico VideoCore IV 3D, que consegue rodar vídeos em 1080p com relativa tranquilidade.[3] A placa possui 4 portas USB, saída de áudio e vídeo composto no mesmo conector, porta HDMI e conectores para câmera e display, além do conector de 40 pinos GPIO.[3] Para o projeto, a Raspberry Pi será utilizada para o processamento de imagens do QrCode feitas pela câmera , e a própria placa acionará o relé e consequentemente a bomba permitindo ou não a utilização da máquina.

Foi escolhida a versão 3 por causa do seu poder de processador de 64 bits, processador gráfico para processamento de imagens, além do seu clock superior a outras versões.

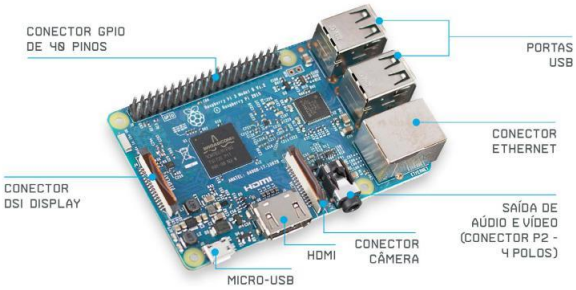

Fig. 2. Placa Raspberry Pi 3.

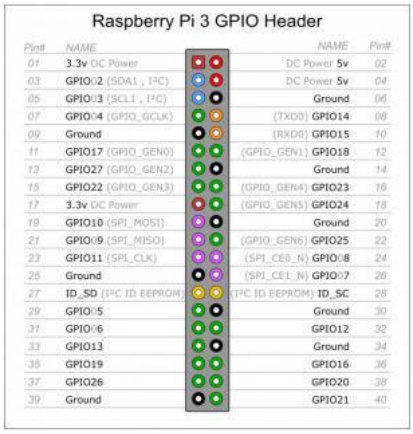

Fig. 3. Pinos GPIO;

5.2. Câmera Raspberry Pi3

A câmera Raspberry Pi é uma câmera digital em um módulo bastante leve, pesando apenas 3 gra-mas, e compacto com medidas de (25 x 20 x 9mm). Ela gera fotos com resolução de até 25921944 pixels e vídeos com resolução de até 1080p.[4]

Fig. 4. Pinos GPIO;

A imagem abaixo apresenta um esquemático mostrando a conexão entre a câmera e a placa Raspberry Pi 3.

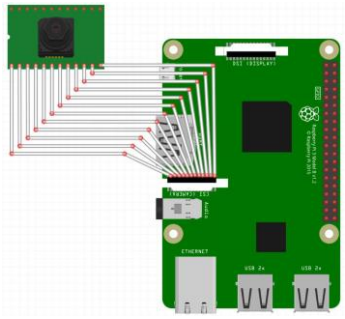

Fig. 5. Conexão entre câmera e a placa.

5.3. Bomba de Aquário

Para realização da máquina de refrigerantes, era necessário encontrar um modo de entrega esse refrigerante ao usuário, e o modo encontrado pela equipe foi de utilizar uma bomba de aquário com o intuito de injetar ar no interior do recipiente contendo liquido, fazendo com que o mesmo seja entregue ao cliente. Para isso foi utilizado a bomba sarlobetter mini a de 2W, que por possuir um baixo consumo e uma vazão de até 1L/min se encaixa no escopo do projeto. As imagens abaixo apresentam uma figura da bomba utilizada no projeto e também um esquemático mostrando o modo que foi feita a conexão entre a bomba e a placa Raspberry Pi 3.

Fig. 6. Bomba de Aquário.

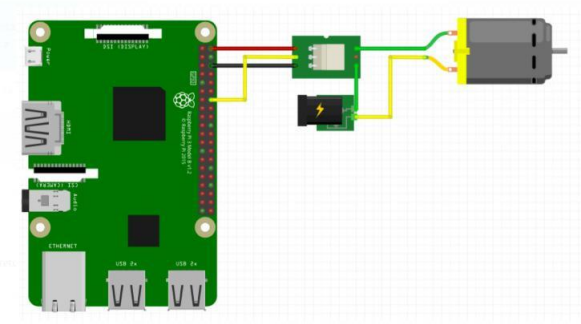

Fig. 7. Conexão entre a bomba e a placa.

5.4. Botão

O botão será utilizado para que quando o qr code for validado, o cliente possa controlar a bomba, ligando-a para que o refrigerante caia no copo, e desligando até o momento que cliente preferir encher. O botão só ativará o relé que ligará a bomba quando o qr code for validado, ou seja esteja dentro do prazo para uso. A conexão do botão se dá entre o GPIO 18 e o relé.

5.5. LEDs

Foram utilizados dois leds no projeto, um verde e um vermelho. O led vermelho ficará aceso enquanto a máquina estiver fechada para uso, e o led vermelho acenderá quando um qr code for validado, mostrando para o usuário que a máquina está pronta para o uso. Foi utilizado um resistor de 220 Ω nas saídas dos pinos.

Tabela 2. Pinagem entre os leds e a placa.

RASPBERRY PI3	LED
GPIO24	Led Vermelho
GPIO23	Led Verde

5.6. Módulo Relé

A utilização do módulo relé vem da necessidade de controle do acionamento da bomba uma vez que a mesma precisa ser alimentada com uma tensão de 220V e a placa trabalha com valores muito inferiores.

Fig. 8. Módulo Relé.

Tabela 3. Pinagem entre o módulo e a placa.

RASPBERRY PI3	MÓDULO RELÉ
PIN 02	VCC
PIN 06	GCC
GPIO18	Botão

Tabela 4. Pinagem entre a bomba e o relé.

MÓDULO RELÉ	BOMBA DE AQUARIO
PIN 01	VCC Bomba
PIN 02	Power

5.7. Circuito Final

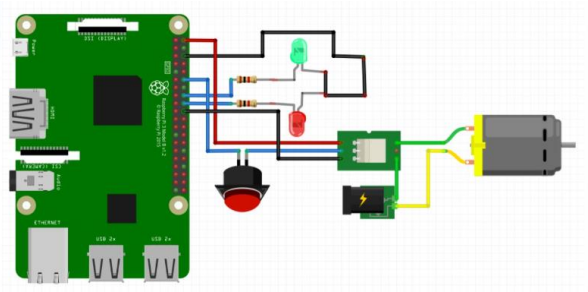

Fig. 9. Circuito Final.

6. ASPECTOS DE SOFTWARE

6.1. Raspberry Pi 3

O sistema operacional utilizado na Raspberry Pi é o Raspbian, sendo este um sistema livre baseado no Debian, otimizado para o hardware Raspberry Pi.[5] Ele foi instalado em um cartão SD para ser utilizado na placa, já que a Raspberry não possui HD interno. Foram criados programas em C e foram compilados pelo gcc através do terminal.

6.2. Câmera

Para utilização da câmera na Raspberry Pi 3 após a conexão de hardware, foi acessada as configurações da placa clicando em Berry -> Preferences -> Raspberry Configuration. Clicando na aba Interfaces, ativamos a câmera clicando em Enabled. Para utilização, será utilizada a ferramenta Ras-pistill. Raspistill é uma ferramenta de comando de linha para capturar imagens da câmera.

raspistill -o imagem.jpg [9]

Pode ser acrescentado o -t para determinar o tempo para a câmera ficar ligada e o -tl para que a câmera tire fotos em um intervalo de tempo. Sabendo disso foi criado um programa onde fotos eram tiradas a uma taxa de 5 fotos por segundo e elas eram sobrepostas umas sobre as outras com o intuído de reduzir a quantidade de memória utilizada pelo sistema. Essas imagens são depois utilizadas para verificação em uma outra rotina dentro do sistema.

6.3. Módulo Relé, Leds

Para utilização do módulo relé e dos leds foram utilizados os pinos de GPIO. A biblioteca utilizada para realização dos códigos em C foi o sysfs. O Sysfs possui a capacidade de permitir ao código do kernel a exportação das informações necessárias para o controle dos periféricos do espaço de núcleo ao espaço do usuário em um sistema de arquivos em memória, que nos possibilita a manipulação do hardware no espaço do usuário mesmo com a presença do sistema operacional. [8]

São 3 códigos em anexo para ativar e desativar o relé. O primeiro nomeado de gpiosysfs.c, trabalha com a coleção de diretórios gerado pela biblioteca. Inicialmente, para que possamos controlar um pino de GPIO, necessariamente precisamos fazer com que este controle feito no espaço de núcleo seja exportado para o espaço de usuário. Após a exportação do pino, precisamos definir como o mesmo deverá trabalhar, se será como OUTPUT ou INPUT através do arquivo direction. Através do arquivo value, é possível realizar a leitura ou escrita no pino de acordo com as configurações definidas no método direction. Por fim, após a o término da utilização do pino exportado, visando evitar que o mesmo esteja consumindo recursos da placa e do S.O, sempre é necessário que um unexport seja

aplicado. [8] Neste código sempre é feita uma verificação de erro nos arquivos para se obter um feedback em relação ao funcionamento.

No segundo código chamado gpiosysfs.h temos o cabeçalho com algumas funções a serem utilizadas para transição e determinação dos dados. No último código, chamado relecomled.c, temos a definição dos pinos de GPIO a serem utilizados, onde nosso caso são os pinos 18 (relé), 23 (led verde) e 24 (led vermelho). Os pinos foram definidos como saída, e posteriormente foram ativados (com exceção do led vermelho). A função sleep para o programa pelo tempo determinado entre parênteses, 10 segundos. No final o pino verde e o pino do relé serão unexportados, ou seja, desligados. Entretanto o led vermelho será ativado para que fique aceso quando a máquina estará inacessível.

6.4. Gerador de QR Code

Para geração de imagens QR code foi utilizada a biblioteca qrencode, podendo ela ser instalada através do comando:

sudo apt-get install qrencode; [7]

Após a instalação da biblioteca para realizar a geração de um código utiliza-se o seguinte comando:

’qrencode "$(Comando)-o Imagem.jpg’ [7]

A utilização do QR-Code se dá pelo armazenamento da data e hora do sistema local, utilizando ele como referência e também o tempo, que pode ser um padrão determinado ou uma variável a ser determinada pelo operador, utilizando esses dois parâmetros é determinado a hora limite de acesso do usuário que utiliza a máquina de refrigerantes. O código se encontra em anexo.

6.5. Verificação de usuário

O código tem como intuito analisar uma imagem recebida e verificar se há ali a presença de um QR-Code. Caso haja alguma informação, ele irá decodificar a mensagem presente e analisar. Uma vez que a mensagem presente no QR-Code é um horário, ocorrerá uma verificação se o mesmo é maior ou menor que o horário atual, fazendo com que seja liberado ou negado o acesso do usuário. O código se encontra em anexo.

7. ESTRUTURA

Para realização do projeto foi pensada uma estrutura que se assemelha as máquinas de café expresso domesticas comercializadas hoje em dia, com o intuído de ser algo simples de se fabricar, mas ao mesmo tempo, que pudesse suprir as necessidades impostas pelos desafios do projeto. Pensando nisso foi feita uma estrutura de acordo com as dimensões da figura a seguir em material MDF de 6mm.

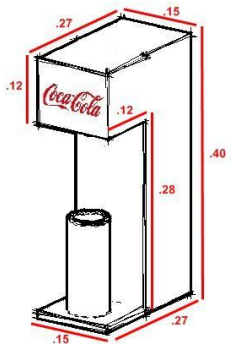

Fig. 10. Dimensionamento da estrutura..

Após a montagem da máquina com os furos e os componentes, se obteve o seguinte resultado:

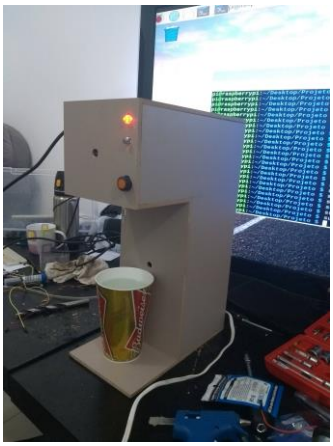

Fig. 10. Estrutura da máquina.

8. BIBLIOGRAFIA

1 Raspberry Pi Cookbook for PythonProgrammers. Cox, Tim.

2 Roubo de Refil. Disponível em http://varelanoticias.com.br/garotos-levam-galao-de-20-litro-para-encher-refil-de-refrigerante-no-burger-king-veja-video

3 Filipe Flop. Guia Raspberry para iniciantes, 2016

4 Câmera Raspberry Pi. Disponível em https://www.filipeflop.com/blog/modulo-camera-raspberry-pi/

5 Welcome to Raspbian. Disponível em https://www.raspbian.org/

6 PiCamera. Disponível em https://projects.raspberrypi.org/en/projects/getting-started-with-picamera

7 Gerar QR codes. Disponível: http://www.dicasl.com.br/arquivo/qrencodeaplicativoparageracoesdecodigosqr.php.W9pFjZNKjIU

8 - Controle GPIO em C. Disponível em https://www.embarc br/gpio-da-raspberry-pi-linguagem-c/

9 – Raspistill. Disponível em: https://www.raspberrypi.org/documentation/usage/camer

9. ANEXOS

9.1 Módulo Relé e LEDS.

```c
#include "gpio_sysfs.h"
#include <unistd.h>

int main(){

        int pin1=18;
        int pin2=23;
        int pin3=24;

        if(setGPIO_Out(pin1))
                return -1;
        if (GPIO_Write(pin1,1))
                return 1;

        if(setGPIO_Out(pin2))
                return -1;
        if (GPIO_Write(pin2,1))
                return 1;

        if(setGPIO_Out(pin3))
                return -1;
        if (GPIO_Write(pin3,0))
                return 1;

        sleep(10);

        if (GPIO_Write(pin3,1))
                return 1;
        if(unsetGPIO(pin1))
                return 2;
        if(unsetGPIO(pin2))
                return 2;
        return 0;
}
```

Fig. 11. Código relecomled.c

```c
extern int setGPIO_Out(int);
extern int GPIO_Write(int, int);
extern int unsetGPIO(int);
```

Fig. 12. Código GPIO gpiosysfs.h.

```c
#define MAXSTR 64
#define VALID_PINS 0, 1, 4, 7, 8, 9, 10, 11, 14, 15, 17, 18, 21, 22, 23, 24, 25

#include <stdio.h>
#include "gpio_sysfs.h"

int setGPIO_Out(int pin)
{
        int valid_pins[]={VALID_PINS};
        int c;
        int valid = 0;

        for (c=0;c<(sizeof(valid_pins) / sizeof(int));c++){
                if(pin == valid_pins[c])
                        valid = 1;
        }
        if(!valid){
                fprintf(stderr, "ERROR: Invalid pin!\nPin %d is not a GPIO pin...\n", pin);
                return -1;
        }

        FILE *sysfs_handle = NULL;

        if ((sysfs_handle = fopen("/sys/class/gpio/export", "w")) == NULL){
                fprintf(stderr, "ERROR: Cannot open GPIO export...\n(Is this program running as root?)\n");
                return 1;
        }

        char str_pin[3];
        snprintf(str_pin, (3*sizeof(char)), "%d", pin);

        if (fwrite(&str_pin, sizeof(char), 3, sysfs_handle)!=3) {
                fprintf(stderr, "ERROR: Unable to export GPIO pin %d\n", pin);
                return 2;
        }
        fclose(sysfs_handle);

        char str_direction_file[MAXSTR];
        snprintf(str_direction_file, (MAXSTR*sizeof(char)), "/sys/class/gpio/gpio%d/direction", pin);
        if ((sysfs_handle = fopen(str_direction_file, "w")) == NULL){
                fprintf(stderr, "ERROR: Cannot open direction file...\n(Is this program running as root?)\n");
                return 3;
         }
         if (fwrite("out", sizeof(char), 4, sysfs_handle) != 4){
                 fprintf(stderr, "ERROR: Unable to write direction for GPIO%d\n", pin);
                 return 4;
         }
         fclose(sysfs_handle);
         return 0;
}

int GPIO_Write(int pin, int value)
{
        if ((value!=0)&&(value!=1)){
                 fprintf(stderr, "ERROR: Invalid value!\nValue must be 0 or 1\n");
                 return -1;
        }

        FILE *sysfs_handle = NULL;
        char str_value_file[MAXSTR];

        snprintf (str_value_file, (MAXSTR*sizeof(char)), "/sys/class/gpio/gpio%d/value", pin);

        if ((sysfs_handle = fopen(str_value_file, "w")) == NULL)
        {
                 fprintf(stderr, "ERROR: Cannot open value file for pin %d...\n(Has the pin been exported?)\
                 return 1;
        }

        char str_val[2];
        snprintf (str_val, (2*sizeof(char)), "%d", value);

        if(fwrite(str_val, sizeof(char), 2, sysfs_handle) != 2)
        {
                fprintf(stderr, "ERROR: Cannot write value %d to GPIO pin %d\n", value, pin);
                return 2;
        }
        fclose(sysfs_handle);

        return 0;
}
```

```c

int unsetGPIO(int pin)
{
        FILE *sysfs_handle = NULL;
        char str_pin[3];
        char str_value_file[MAXSTR];

        snprintf (str_pin, (3*sizeof(char)), "%d", pin);
        snprintf (str_value_file, (MAXSTR*sizeof(char)), "/sys/class/gpio/gpio%d/valu

        if ((sysfs_handle = fopen(str_value_file, "w")) == NULL){
                fprintf(stderr, "ERROR: Cannot open value file for pin %d...\n", pin)
                return 1;
        }

        if(fwrite("0", sizeof(char), 2, sysfs_handle) != 2){
                fprintf(stderr, "ERROR: Cannot write to GPIO pin %d\n", pin);
                return 2;
        }
        fclose(sysfs_handle);

        if ((sysfs_handle = fopen("/sys/class/gpio/unexport", "w")) == NULL){
                fprintf(stderr, "ERROR: Cannot open GPIO unexport...\n");
                return 1;
        }

        if (fwrite(&str_pin, sizeof(char), 3, sysfs_handle)!=3) {
                fprintf(stderr, "ERROR: Unable to unexport GPIO pin %d\n", pin);
                return 2;
        }
        fclose(sysfs_handle);
        return 0;
}
```

Fig. 13. Código GPIO gpiosysfs.c

9.2 Cadastro do usuário

```
while true
do
read -p "Informe o tempo de acesso do cliente em minutos: " MIN
CAL=$(date +%M)         #Declaracao da variavel minuto do calendario
MIN=$(($MIN+$CAL))      #Minuto 'e igual ao minuto do calendario + minutos inseriods p
HORA=$(date +%H)        #Hora do calendario
DIA=$(date +%H)                 #Dia do calendario


while test "$MIN" -gt 60        #Em quanto min for maior que 60 faca
do                              #While transforma o credito inserido pelo
        if test "$MIN" -gt 60   #usuario em um horario futuro
        then
                HORA=$(($HORA+1))
                MIN=$(($MIN-60))

        fi
done


                MIN=$(($MIN+100))
                MIN=$(echo "${MIN#1}")
                HORA=$(($HORA+100))
                HORA=$(echo "${HORA#1}")
echo "O tempo limite de acesso do cliente é $HORA:$MIN"
$(qrencode " $HORA:$MIN" -o QRCode.png)         #Insere no QR code o horario de
                                        #se encerra o tempo do usuario

done
```

Fig. 14. Código de gerar Qr code.

9.2 Validação do usuário

```

while true
do

#!/bin/bash
# $(raspistill -t 1 -o QRCode.png)
QR=$(zbarimg -Sdisable -Sqrcode.enable -q QRCode.png)
TE=$(echo $?)
QR1=$(echo "${QR#QR-Code: }")   #Seleciona somento o horario dentro da variavel
MIN1=$(echo "${QR1:3}")         #Minutos do QRcode
HR1=$(echo "${QR1:0:2}")        #Horas do QRCode
MIN=$(date +%M)                 #Minutos do Relogio
HR=$(date +%H)                  #Horas do Relogio
HORA=$(($HR1-$HR))
if test "$TE" -ne 4
        then
if test "$MIN1" -lt "$MIN" -a "$HORA" -gt 1
        then
                MIN1=$(($MIN1+60))
                HR1=$(($HR1-1))
        fi
MIN1=$(echo "${MIN1#0}")
MINUTO=$(($MIN1-$MIN))
HORA=$(($HR1-$HR))
#MINUTO=$(($MINUTO+100))
#MINUTO=$(echo "${MINUTO#1}")

echo "Seu tempo restante é de $HORA Horas e $MINUTO Minutos"
case $HORA in
 [1-9])
        echo "Aberto"
        $(sudo ./relecomled)
        ;;
-[1-9])
        echo "Fechado"
        ;;
 0)
        case $MINUTO in
                0)
                echo "Fechado"
                ;;
                [0-6][0-9])
                echo "Aberto"
                $(sudo ./relecomled)
                ;;
                *)
                echo "Fechado"
                ;;
        esac

esac

fi
done
```

Fig. 16. Código para Validação do QR-Code.