

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, hiper-mercados, grandes centros de comércio 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 burlam 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 bomba liberando o funcionamento.

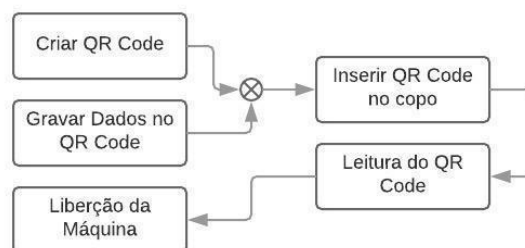


Fig. 1. Fluxograma do projeto.

4. BENEFÍCIOS

O intuito da realização deste projeto está em conseguir que a utilização de um produto seja dada so-mente pelos clientes da loja de alimentos, fazendo com que haja a diminuição ou até mesmo a extin-ção 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 projetos, 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

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 Video-Core 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, pro-cessador

gráfico para processamento de imagens, além do seu clock superior a outras versões.

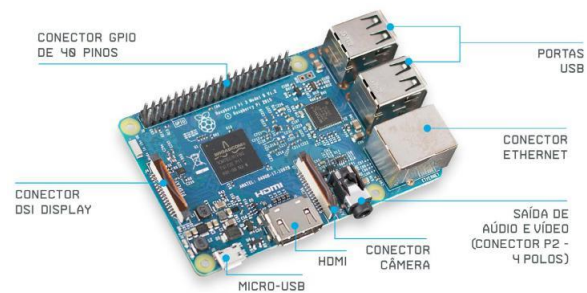


Fig. 2. Placa Raspberry Pi 3.

Raspberry Pi 3 GPIO Header			
Pin#	NAME		NAME Pin#
01	3.3v DC Power		DC Power 5v 02
03	GPIO2 (SDA1, I2C)		DC Power 5v 04
05	GPIO3 (SCL1, I2C)		Ground 06
07	GPIO4 (GPIO_GCLK)		(TXD0) GPIO14 08
09	Ground		(RXD0) GPIO15 10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18 12
13	GPIO27 (GPIO_GEN2)		Ground 14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23 16
17	3.3v DC Power		(GPIO_GEN5) GPIO24 18
19	GPIO10 (SPI_MOSI)		Ground 20
21	GPIO9 (SPI_MISO)		(GPIO_GEN6) GPIO25 22
23	GPIO11 (SPI_CLK)		(SPI_CEO_N) GPIO28 24
25	Ground		(SPI_CEL_N) GPIO27 26
27	ID_SD (I2C ID EEPROM)		(I2C ID EEPROM) ID_SC 28
29	GPIO5		Ground 30
31	GPIO6		GPIO12 32
33	GPIO13		Ground 34
35	GPIO19		GPIO16 36
37	GPIO26		GPIO20 38
39	Ground		GPIO21 40

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 gramas, e compacto com medidas de (25 x 20 x 9mm). Ela gera fotos com resolução de até 2592x1944 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.

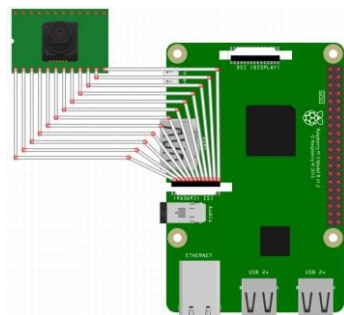


Fig. 5. Conexão entre câmera e a placa.

5.3. Bomba de Aquário

Para realização da máquina de refrigerantes, era necessário encontrar um modo de entrega desse re-frigerante 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 con-tendo liquido, fazendo com que o mesmo seja en-tregue ao cliente. Para isso foi utilizado a bomba sarlobetter mimi 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 tam-bém um esquemático mostrando o modo que foi feita a conexão entre a bomba e a placa Raspberry Pi 3.



Fig. 6. Bomba de Aquário.

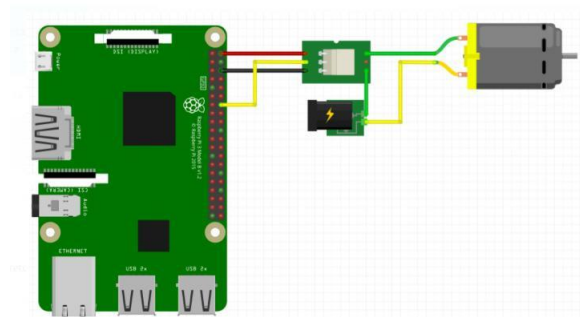


Fig. 7. Conexão entre a bomba e a placa.

Tabela 2. Pinagem entre o módulo e a placa.

RASPBERRY PI3	MÓDULO RELÉ
PIN 02	VCC
PIN 06	GCC
PIN 12	Signal

5.4. 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 ten-são de 220V e a placa trabalha com valores muito inferiores.



Fig. 8. Módulo Relé.

Tabela 3. Pinagem entre a bomba e o relé.

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

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 Raspistill. 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, dada os dois em milissegundos.

6.3. Módulo Relé

Para utilização do módulo relé, 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 rele.c, temos a definição do gpio a ser utilizado, onde nosso caso é o 18. O pino foi definido como saída, e posteriormente foi ativado, para ativação do relé. A função sleep para o programa pelo tempo de determinado entre parênteses, 5 segundos, e o pino é unexportado para que sua utilização seja desligada.

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]
```

Sabendo disso, o intuito do código em QR é guardar o tempo limite de uso da máquina permitida para o cliente. 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 intuito 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.

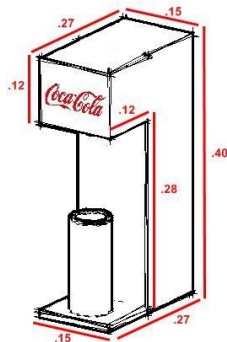


Fig. 9. Dimensionamento Estrutura.



Fig. 10. Estrutura pronta 1.

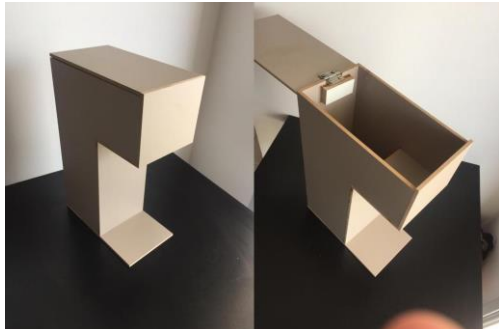


Fig. 11. Estrutura pronta 2.

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/camera/>

9. ANEXOS

9.1 Módulo Relé/Bomba

```
1  #include "gpio_sysfs.h"
2  #include <unistd.h>
3
4  int main()
5  {
6      int pin=18;
7      if(setGPIO_Out(pin))
8          return -1;
9      if (GPIO_Write(pin,1))
10         return 1;
11     sleep(5);
12     if(unsetGPIO(pin))
13         return 2;
14     return 0;
15 }
```

Fig. 12. Código relé.

```

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

```

Fig. 13. Código GPIO gpiosysfs.h.

```

1  #define MAXSTR 64
2  #define VALID_PINS 0, 1, 4, 7, 8, 9, 10, 11, 14, 15, 17, 18, 21, 22, 23, 24, 25
3
4  #include <stdio.h>
5  #include "gpio_sysfs.h"
6
7  int setGPIO_Out(int pin)
8  {
9      int valid_pins[]={VALID_PINS};
10     int c;
11     int valid = 0;
12
13     for (c=0;c<(sizeof(valid_pins) / sizeof(int));c++){
14         if(pin == valid_pins[c])
15             valid = 1;
16     }
17     if(!valid){
18         fprintf(stderr, "ERROR: Invalid pin!\nPin %d is not a GPIO pin...\n", pin);
19         return -1;
20     }
21
22     FILE *sysfs_handle = NULL;
23
24     if ((sysfs_handle = fopen("/sys/class/gpio/export", "w")) == NULL){
25         fprintf(stderr, "ERROR: Cannot open GPIO export...\n(Is this program running as root?)\n");
26         return 1;
27     }
28
29     char str_pin[3];
30     snprintf(str_pin, (3*sizeof(char)), "%d", pin);
31
32     if (fwrite(&str_pin, sizeof(char), 3, sysfs_handle)!=3) {
33         fprintf(stderr, "ERROR: Unable to export GPIO pin %d\n", pin);
34         return 2;
35     }
36     fclose(sysfs_handle);
37
38     char str_direction_file[MAXSTR];
39     snprintf(str_direction_file, (MAXSTR*sizeof(char)), "/sys/class/gpio/gpio%d/direction", pin);
40     if ((sysfs_handle = fopen(str_direction_file, "w")) == NULL){
41         fprintf(stderr, "ERROR: Cannot open direction file...\n(Is this program running as root?)\n");
42         return 3;
43     }
44     if (fwrite("out", sizeof(char), 4, sysfs_handle) != 4){
45         fprintf(stderr, "ERROR: Unable to write direction for GPIO%d\n", pin);
46         return 4;
47     }
48     fclose(sysfs_handle);
49     return 0;
50 }
51
52 int GPIO_Write(int pin, int value)
53 {
54     if ((value!=0)&&(value!=1)){
55         fprintf(stderr, "ERROR: Invalid value!\nValue must be 0 or 1\n");
56         return -1;
57     }
58
59     FILE *sysfs_handle = NULL;
60     char str_value_file[MAXSTR];
61
62     snprintf (str_value_file, (MAXSTR*sizeof(char)), "/sys/class/gpio/gpio%d/value", pin);
63
64     if ((sysfs_handle = fopen(str_value_file, "w")) == NULL)
65     {
66         fprintf(stderr, "ERROR: Cannot open value file for pin %d...\n(Has the pin been exported?)\n");
67         return 1;
68     }
69
70     char str_val[2];
71     snprintf (str_val, (2*sizeof(char)), "%d", value);
72
73     if(fwrite(str_val, sizeof(char), 2, sysfs_handle) != 2)
74     {
75         fprintf(stderr, "ERROR: Cannot write value %d to GPIO pin %d\n", value, pin);
76         return 2;
77     }
78     fclose(sysfs_handle);
79
80     return 0;
81 }

```

```

82
83 int unsetGPIO(int pin)
84 {
85     FILE *sysfs_handle = NULL;
86     char str_pin[3];
87     char str_value_file[MAXSTR];
88
89     snprintf (str_pin, (3*sizeof(char)), "%d", pin);
90     snprintf (str_value_file, (MAXSTR*sizeof(char)), "/sys/class/gpio/gpio%d/value", pin);
91
92     if ((sysfs_handle = fopen(str_value_file, "w")) == NULL){
93         fprintf(stderr, "ERROR: Cannot open value file for pin %d...\n", pin);
94         return 1;
95     }
96
97     if(fwrite("0", sizeof(char), 2, sysfs_handle) != 2){
98         fprintf(stderr, "ERROR: Cannot write to GPIO pin %d\n", pin);
99         return 2;
100    }
101    fclose(sysfs_handle);
102
103    if ((sysfs_handle = fopen("/sys/class/gpio/unexport", "w")) == NULL){
104        fprintf(stderr, "ERROR: Cannot open GPIO unexport...\n");
105        return 1;
106    }
107
108    if (fwrite(&str_pin, sizeof(char), 3, sysfs_handle)!=3) {
109        fprintf(stderr, "ERROR: Unable to unexport GPIO pin %d\n", pin);
110        return 2;
111    }
112    fclose(sysfs_handle);
113    return 0;
114 }

```

Fig. 14. Código GPIO gpiosysfs.c

9.2 Cadastro do usuário

```

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
HORA=$(date +%H)         #Hora do calendario
DIA=$(date +%H)          #Dia do calendario

while test "$MIN" -gt 60      #Em quanto min for maior que 60 faca
do
    if test "$MIN" -gt 60
    then
        HORA=$(( $HORA+1))
        MIN=$(( $MIN-60))
        if test "$MIN" -lt 10
        then
            MIN = "0$MIN"
        fi
    fi
done

MIN=$(( $MIN+100))
MIN=$(echo "$MIN#1")

echo "O tempo limite de acesso do cliente é $HORA:$MIN"
$(qrencode " $HORA:$MIN" -o QRCode.png)

```

Fig. 15. Código para gerar Qr-Code.

9.2 Validação do usuário

```
1  QR=$(zbarimg QRCode.png)
2  QR1=$(echo "${QR#QR-Code: }") #Seleciona somente o horario dentro da variavel
3  MIN1=$(echo "${QR1:3}")        #Minutos do QRcode
4  HR1=$(echo "${QR1:0:2}")       #Horas do QRCode
5  MIN=$(date +%M)               #Minutos do Relogio
6  HR=$(date +%H)               #Horas do Relogio
7  HORA=$((HR1-HR))              #Diferenca entre Hora do usuario e horario atual
8  MINUTO=$((MIN1-MIN))          #Diferenca entre minuto do usuario e o minuto atual
9
10 case $MINUTO in
11   -[0-5][0-9])                #Caso a diferenca seja negativa
12     HORA=$((HORA-1))           #Diminui uma hora
13     MINUTO=$((MINUTO+60))      #Acrescenta 60 minutos
14   esac
15
16 echo "Seu tempo restante é de $HORA Horas e $MINUTO Minutos"
17 $case $HORA in
18   [1-9])                      #Case relacionado a variavel Hora
19     echo "Open"                #Caso o usuario possua um saldo
20     ;;                         #possitivo de horas o acesso é
21   -[1-9])                      #liberado
22     echo "Closed"              #Caso o saldo seja negativo o acesso
23     ;;                         # é negado
24   0)
25     case $MINUTO in
26       0)                       #Caso o saldo seja nulo é feita
27         echo "Closed"          #a mesma analise para os minutos
28         ;;
29         [0-6][0-9])
30         echo "Open"
31         ;;
32         *)
33         echo "Closed"
34         ;;
35     esac
36   esac
37
38
```

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