MaskBerry

Sistema para verificação do uso de máscara e controle do fluxo de pessoas em uma padaria utilizando Raspberry Pi

Luísa Caroline Alves Silva 16/0134587
Universidade de Brasília
Faculdade Gama
Brasília, Brasil
luisacarollinne@gmail.com

Manuella Cristina Panza Ramos 16/0135290
Universidade de Brasília
Faculdade Gama
Brasília, Brasil
manuellapanza@gmail.com

***Resumo*—A transmissão do novo coronavírus costuma ser pelo ar ou por secreções, com isso é importante que o número de pessoas em diferentes locais (principalmente fechados) seja controlado e o espaço devidamente higienizado, além do uso de máscara, pois assim evita o aumento de chances de contaminação. Neste caso, o presente trabalho propõe a implementação de um sistema controlado a partir do reconhecimento facial para monitorar o uso de máscara, automação de um dispenser de álcool em gel, impressão automática de senha e contagem de pessoas presentes no local para controle de superlotação. Ao detectar algum tipo de infração, o sistema irá se comunicar com o usuário (funcionário) por meio de um bot do Telegram.**

***Index Terms*—Raspberry pi, Reconhecimento Facial, Automação, COVID-19.**

I. Justificativa

Em consequência da pandemia do novo coronavírus, o monitoramento na entrada do estabelecimentos é essencial para o controle de fluxo de pessoas e verificação do uso obrigatório de máscara. Muitos locais fazem tal monitoramento pela escalação de funcionários, o que ocorre a exposição destes e dos clientes a um maior risco de contágio. Além disso, o controle de quantidade total pessoas dentro do estabelecimento, na maioria das vezes, não é bem contabilizada, e pode assim, causar uma grande aglomeração no local.

II. Objetivos

O objetivo é projetar um sistema capaz de determinar se a pessoa está usando máscara facial ou não. Assim, ao realizar a detecção do rosto por meio de uma câmera e capturar a imagem para análise de dados, pode-se identificar entradas não autorizadas de pessoas sem máscara de proteção. O intuito é alertar ao gerente este tipo de entrada e, consequentemente atentar ao indivíduo o risco de possível propagação e contágio do vírus que ela poderá ser submetida. Este projeto conta também com a intenção de monitorar e informar a quantidade de pessoas no estabelecimento para prevenir aglomeração, além de um dispensar álcool em gel para higienização.

III. Requisitos

Para realização do projeto precisa-se de:

- Uma Raspberry Pi onde funcionará como unidade central, em que ocorrerá o processamentos de dados adquiridos, além de passar a informações para clientes se o mesmo pode ou não entrar no local e informar ao gerente do local quando algum cliente entrar sem máscara;
- Uma câmera que irá capturar a imagem das pessoas entrando em determinado local para detectar se a pessoa usa máscara facial ou não;
- Sensores de distância pra realizar a contagem de entrada e saída de pessoas do estabelecimento e com o mesmo modelo de sensor, também vai ser implementado um dispenser de álcool na a entrada do local;
- Display LCD para exibir o uso de máscara e informar se o mesmo pode ou não entrar no estabelecimento;
- Displays de 7 segmentos que mostrará a quantidade de pessoas dentro do local de acordo com a contagem realizada pelos sensores;
- Um bot no Telegram para informar ao gerente se algum sujeito foi identificado com uma entrada não autorizada;
- O circuito eletrônico montado será totalmente lacrado para não haver alterações ou desconexão elétrica. O projeto é designado para locais pequenos, nesse caso, uma padaria.

IV. Benefícios

Uma vez que a utilização de máscaras são essenciais para proteger e evitar a propagação do COVID-19, este projeto pode ser utilizado em hospitais, bancos, mercados, e outros encontros públicos onde o monitoramento deve ser feito. Dessa forma, é possível ter um determinado controle e decisões mais assertivas ao gerente sobre os procedimentos de prevenção e normas de segurança ao coronavírus em estabelecimentos e, para os indivíduos, em relação à segurança ao visitar locais com maior movimentação de pessoas e evitar o contato direto com objetos.

V. Revisão Bibliográfica

Diante do projeto proposto, foi realizado um estudo em aplicações de reconhecimento facial, que, apesar de ser usado para analisar características faciais [1], vai ser utilizado de forma reversa, ou seja, uma lógica negativa a fim de detectar o uso de máscara pelo não reconhecimento de características completas do rosto. Uma forma de impedir a propagação do COVID-19 é evitar aglomerações e a aproximação entre as pessoas, além disso, o cuidado com a higiene deve ser ainda mais intenso durante a pandemia [2]. Assim, foi encontrado

dados capazes de implementar um protótipo que verifica o cumprimento de normas de segurança à saúde ao coronavírus em relação ao uso de máscara [3][4]. Uma adaptação prevista para o projeto é automação de processos para evitar o contágio por meio de contato [5], certificar e informar o risco de aglomeração através de avisos visuais como uma medida preventiva da propagação do vírus.

VI. Desenvolvimento

A partir do desenvolvimento dos requisitos podemos metrificar o andamento do projeto rumo aos objetivos.

A. *Descrição de Hardware*

1) *Lista de materiais*:
   - Raspberry Pi 3B
   - Servo Motor SG 90
   - Sensor de Obstáculo Infravermelho
   - Display de 7 segmentos
   - Display LCD 16x2
   - Decodificador BCD 7 segmentos
   - Reservatório para álcool
   - WebCam
   - Impressora térmica
2) *Utilização dos componentes*: Para cumprir o requisito de contagem de entrada e saída de indivíduos no estabelecimento, foi projetado um hardware capaz de realizar a detecção de entrada e saída de pessoas utilizando sensores de obstáculo infravermelho (IR) e exibir a contagem em um display de 7 segmentos (integrado a um decodificador) para mostrar o valor numérico da quantidade total de clientes no local. Ao identificar a presença de algum objeto (no nosso caso, uma pessoa) a saída do sensor vai para nível lógico baixo, contabilizando a entrada ou saída de uma pessoa, e quando não entra e não sai alguém o nível lógico permanece em alto.

   De maneira semelhante, utilizando outros sensores de obstáculo do mesmo modelo citado, foi projetado um sistema de dispensação de álcool automática e impressão de senha, que ao aproximar a mão de um desses sensores, ativa o servomotor e libera álcool e o outro imprime a senha.

   Para realização da análise do uso de máscara, será utilizado uma WebCam para captura da imagem, que será processado por um software. E por meio do resultado do processamento e contagem de pessoas no local, é exibida mensagens através do display LCD informando se a pessoa usa ou não máscara e se o local está superlotado. Pode-se visualizar na Figura 1 o esquemático representativo do hardware montado.

B. *Descrição de Software*

A ideia geral da explicação do funcionamento de software descrito neste ponto de controle pode ser observado, com base no fluxograma da Figura 2, que pretende ser seguido para as implementações futuras.

1) *Contagem de pessoas*: Utilizou-se 2 sensores IR, um para a contagem de entrada que serve para incrementar uma variável $qtd\_pessoas$ e um para saída, que serve para decrementar a mesma variável e indicar a retirada da pessoa no local. Assim, o valor da variável é sempre atualizado em dois displays de 7 segmentos, em que, um é pra unidade e outro é pra dezena. Como o estabelecimento não pode ter um número grande de pessoas no seu interior foi definido utilizar somente a ideia de dezena e unidade.
2) *Dispenser e Impressora térmica*: Com a finalidade de atender ao requisito do dispenser de álcool em gel, foi proposto o uso do sensor IR que, se o dado obtido da sua saída for nível lógico baixo, é implementado uma lógica para comandar o movimento do servomotor por PWM, se a saída não for de nível baixo, ele não recebe o comando. Ao ativar o servomotor ele controlará a descida da válvula de álcool para a mão da pessoa. Já a impressora térmica terá uma comunicação com a Raspberry Pi por meio de WiFi, e funcionará a partir do

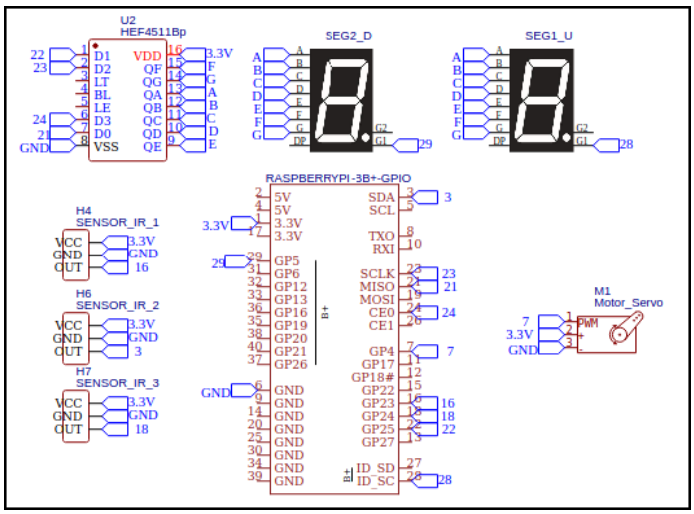

Figura 1. Esquemático do hardware do sensor de obstáculo IR e display LED

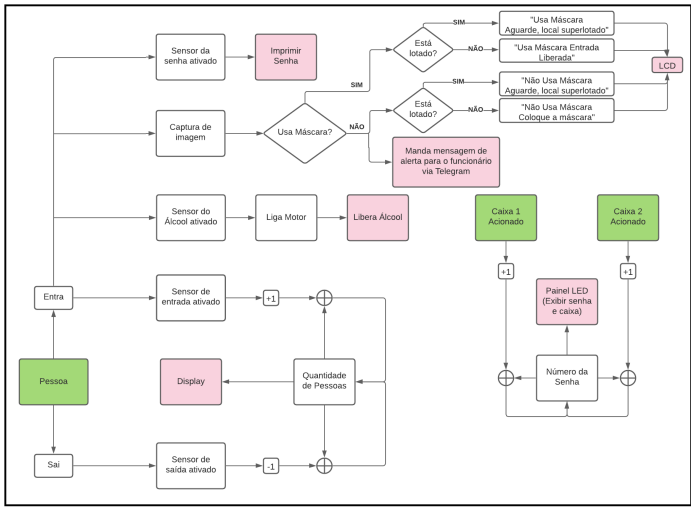

Figura 2. Fluxograma

momento em que é uma variável $senha$ é incrementada (quando detectado um sinal de presença por meio do sensor) e assim imprimir o valor da mesma.

3) *Detecção de Face*: A detecção de face constitui a primeira necessidade do projeto. Foi pesquisado um método de implementação por meio de classificadores em cascata de Haar implementados por meio da biblioteca de visão computacional de tempo real OpenCV. Os pontos de referência faciais consiste em inferir automaticamente a localização das estruturas faciais, incluindo: olhos, sobrancelha, nariz, boca, mandíbula. Logo, o método a ser utilizado para identificar o uso de máscara reflete na lógica negativa dessas referências faciais, pelo não reconhecimento de características completas do rosto. Para a presente etapa do projeto esse requisito está sendo implementado e requer testes posteriores. Ademais, o sistema permite somente a utilização de um usuário por vez, filtrando a maior face presente no vídeo obtido.

VII. Resultados

Visando os objetivos do Ponto de Controle 2, fez-se testes separadamente para cada um dos subsistemas, com intuito de efetuar uma validação prática destes. Em particular, para validação da utilização dos componentes que conseguiu-se adquirir até o momento, foi construído o esquemático mostrado na Figura 1 e implementado esse sistema na RaspBerry Pi 3B.

Foi levado em consideração a passagem da pessoa e sua possível permanência na frente do sensor para que não houvesse um incremento repetitivo, visto que, neste caso ele estaria sempre com o nível lógico baixo indicando a presença de pessoa. Com isso o sistema não fica contando indevidamente a quantidade de pessoas no estabelecimento.

A utilização de multi-processos nessa etapa foi crucial para a execução do sistema ainda parcialmente implementado, dado que as tarefas aqui relatadas estavam sendo executadas todas quase simultaneamente, graças a implementação de processos pai e filho.

VIII. Referências

[1] Face Recognition with OpenCV. Disponível em: <https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html> Acesso em: 17 de set. 2020.

[2] de Oliveira, TM. Manifestações e aglomerações em períodos de pandemia por COVID-19 Manifestações em períodos de pandemia. InterAm J Med Health 2020;3:e202003025.(COVID-19). InterAm J Med Health 2020;3:e202003024.

[3] COVID-19 - Authorized Entry Using Face Mask Detection, 2020. Disponível em: <https://www.hackster.io/rahulkhanna/covid-19-authorized-entry-using-face-mask-detection-37ad77>. Acesso em: 14 de set. 2020.

[4] Smart Gateway Screening for Covid 19, 2020. Disponível em: <https://www.hackster.io/dadanugm07/smart-gateway-screening-for-covid-19-568d59>. Acesso em: 14 de set. 2020.

[5] Como a automação industrial pode ajudar, 2020. Disponível em: <https://www.arvsystems.com.br/como-a-automacao-industrial-pode-ajudar>. Acesso em: 17 de set. 2020.

ANEXOS

Listing 1. Código principal

```
#include<stdio.h>
#include<wiringPi.h>
#include"decod.h"
#include<sys/types.h>
#include <unistd.h>

#define SENSOR_ENTRADA 0
#define SENSOR_SAIDA 2
#define SENSOR_ALCOOL 3
#define LED 7
#define CATODO_D 28
#define CATODO_U 29

int qtd_pessoas = 0;  // entrada = 0,
    saida = 0;


int main()
{
    wiringPiSetup();
    pinMode(SENSOR_ENTRADA,INPUT);
    pinMode(SENSOR_SAIDA,INPUT);
    pinMode(SENSOR_ALCOOL,INPUT);
    pinMode(LED,OUTPUT);
    pinMode(CATODO_U,OUTPUT);
    pinMode(CATODO_D,OUTPUT);
    pid_t pid1;

    pid1 = fork();

    //filho cuida do sensor e o pai cuida
        do alcool
    if (pid1 == 0)
    {
    while(1)
    {
        if(digitalRead(SENSOR_ENTRADA) ==
            0)          // detecta se h
           presen a
        {
        if (qtd_pessoas<100 &&
           qtd_pessoas>=0)
        {
            qtd_pessoas=qtd_pessoas+1;
            printf("%d\n", qtd_pessoas);
        }

        while(digitalRead(SENSOR_ENTRADA)
            == 0)
        {
            decode(qtd_pessoas%10);
            digitalWrite(CATODO_U,HIGH);
            digitalWrite(CATODO_D,LOW);
            usleep(5000);
            decode(qtd_pessoas/10);
            digitalWrite(CATODO_U,LOW);
            digitalWrite(CATODO_D,HIGH);
            usleep(5000);
            }
        // aguarda enquato o sensor ainda
            esta com presen a
        }else if(digitalRead(SENSOR_SAIDA
            ) == 0)          // detecta se
            h   presen a
         {
        if (qtd_pessoas>0)
        {
          qtd_pessoas= qtd_pessoas-1;
            printf("%d\n", qtd_pessoas);
        }

        while(digitalRead(SENSOR_SAIDA)
            == 0)
        {
            decode(qtd_pessoas%10);
            digitalWrite(CATODO_U,HIGH);
            digitalWrite(CATODO_D,LOW);
            usleep(5000);
            decode(qtd_pessoas/10);
            digitalWrite(CATODO_U,LOW);
            digitalWrite(CATODO_D,HIGH);
            usleep(5000);
            } // aguarda enquato o sensor
                ainda esta com presen a
         }


        decode(qtd_pessoas%10);
        digitalWrite(CATODO_U,HIGH);
        digitalWrite(CATODO_D,LOW);
        usleep(5000);
        decode(qtd_pessoas/10);
        digitalWrite(CATODO_U,LOW);
        digitalWrite(CATODO_D,HIGH);
        usleep(5000);

    }

    }else{

         while(1){

          if(digitalRead(SENSOR_ALCOOL)
              == 0)          // detecta se
               h   presen a
          {
            digitalWrite(LED,HIGH);
        usleep(500000);
```

```
            while(digitalRead(
                SENSOR_ALCOOL) == 0); //
                aguarda enquato o sensor
                ainda esta com presen a
            digitalWrite(LED,LOW);
        }
    }
    }

return 0;
}
```

Listing 2. Decodificador

```
#include<stdio.h>
#include<wiringPi.h>
#include"decod.h"

#define A 21
#define B 22
#define C 23
#define D 24

void decode(int qtd_pessoas)
{
    wiringPiSetup;
    pinMode(A,OUTPUT);
    pinMode(B,OUTPUT);
    pinMode(C,OUTPUT);
    pinMode(D,OUTPUT);

    switch (qtd_pessoas)
    {
    case 0:
        digitalWrite(A,LOW);
        digitalWrite(B,LOW);
        digitalWrite(C,LOW);
        digitalWrite(D,LOW);
        break;
    case 1:
        digitalWrite(A,HIGH);
        digitalWrite(B,LOW);
        digitalWrite(C,LOW);
        digitalWrite(D,LOW);
        break;
    case 2:
        digitalWrite(A,LOW);
        digitalWrite(B,HIGH);
        digitalWrite(C,LOW);
        digitalWrite(D,LOW);
        break;
    case 3:
        digitalWrite(A,HIGH);
        digitalWrite(B,HIGH);
        digitalWrite(C,LOW);
        digitalWrite(D,LOW);
        break;
    case 4:
        digitalWrite(A,LOW);
        digitalWrite(B,LOW);
        digitalWrite(C,HIGH);
        digitalWrite(D,LOW);
        break;
    case 5:
        digitalWrite(A,HIGH);
        digitalWrite(B,LOW);
        digitalWrite(C,HIGH);
        digitalWrite(D,LOW);
        break;
    case 6:
        digitalWrite(A,LOW);
        digitalWrite(B,HIGH);
        digitalWrite(C,HIGH);
        digitalWrite(D,LOW);
        break;
    case 7:
        digitalWrite(A,HIGH);
        digitalWrite(B,HIGH);
        digitalWrite(C,HIGH);
        digitalWrite(D,LOW);
        break;
    case 8:
        digitalWrite(A,LOW);
        digitalWrite(B,LOW);
        digitalWrite(C,LOW);
        digitalWrite(D,HIGH);
        break;
    case 9:
        digitalWrite(A,HIGH);
        digitalWrite(B,LOW);
        digitalWrite(C,LOW);
        digitalWrite(D,HIGH);
        break;

    default:
        break;
}
}
```

Listing 3. Makefile

```
projeto: teste.o decod.o
    gcc $(CFLAGS) -o teste teste.o decod.
        o -lwiringPi
teste.o: teste.c decod.h
    gcc $(CFLAGS) -c teste.c -lwiringPi
decod.o: decod.c decod.h
    gcc $(CFLAGS) -c decod.c -lwiringPi
clean:
    rm -f *.o teste
```