Projeto para Introdução de Engenharia Mecatrônica: Supermercado automatizado durante os tempos de pandemia

Raul Myron Silva Amorim
200049712@aluno.unb.br
20/0049712

Vinícius Carvalho Lima Alcanfor
200044231@aluno.unb.br
20/0044231

Wallace Ben Teng Lin Wu
200028880@aluno.unb.br
20/0028880

Lucas Ramson Siefert
200023047@aluno.unb.br
20/0023047

Hiago dos Reis Leite
190125799@aluno.unb.br
19/0125799

Pedro Dantas Freitas
200026011@aluno.unb.br
20/0026011

Abstract

Modern technology has been a great ally in the combat against the SARS-COV 19 pandemic, like so this introduction to mechatronic engineering project was developed aiming at the creation of new technology capable of minimizing the contamination rates of the virus. This project develops three arduino based systems that will function in close environments such as supermarkets. Each system is responsible for preventing contamination in a specific manner: by limiting the people per area ratio inside the establishment, sanitizing clients and workers hands and ensuring the minimum social distancing.

The first utilizes lasers and sensors to count the amount of people entering, inside and coming out of the establishment in order to maintain a maximum people per square meter ratio; The second uses sensors e motors to automatically dispense alcohol into the user's hands; And the third utilizes distance meters to ensure that two individuals do not surpass the minimum safety distance between each other. With that being said, each system ensures the compliance of the basic safety measures, making the supermarket a safer place.

Keywords: *Supermarket, Arduino, Automatization, Coronavirus, Technology, Mechatronics, Social Distancing, Preventive Measures.*

Resumo

A tecnologia tem sido uma importante aliada no combate à pandemia do SARS-COV 19, dessa forma esse projeto de introdução a engenharia mecatrônica foi desenvolvido com a finalidade de utilizar a tecnologia disponível no intuito de minimizar a contaminação pelo vírus. Neste projeto foram desenvolvidos três sistemas de automação, dos quais têm o arduino como motherboard e atuam em ambientes fechados, tais como supermercados. Sendo cada um desses sistemas responsável por prevenir a contaminação de forma específica, a partir da limitação da densidade populacional do estabelecimento, higienização das mãos de clientes e funcionários, além de garantir o distanciamento social.

A primeira utiliza um par de lasers e de sensores para contabilizar o número de pessoas entrando, dentro e saindo do estabelecimento visando manter um número máximo de clientes por metro quadrado, pode também possivelmente funcionar em corredores; A segunda utiliza sensores e atuadores para automaticamente dispensar álcool nas mãos dos usuários, garantindo sua higienização; E a terceira utiliza medidores de distância para garantir que duas pessoas não transgridem a distância mínima de segurança. Dessa forma, cada sistema garante o respeito às normas básicas de segurança na pandemia, tornando o supermercado um ambiente mais seguro.

Palavras-chave: *Supermercado, Arduino, Automação, Coronavírus, Tecnologia, Mecatrônica, Distanciamento Social, Medidas Preventivas.*

1. Introdução

No ano de 2020, acompanhou-se as múltiplas fases da pandemia do vírus SARS-COV 19 que afetou globalmente a economia e as relações humanas. Em meio a crise mundial, surgiu-se a necessidade de soluções práticas para diminuir a disseminação massiva do vírus e evitar uma crise generalizada nos hospitais e postos de saúde. Nesse sentido, o desenvolvimento científico e tecnológico tem sido uma vanguarda contra o COVID-19 [1]

Sabendo que um possível foco de contágio é o supermercado, visto sua essencialidade na vida cotidiana e alto fluxo de pessoas, torna-se extremamente necessário medidas preventivas nesses locais. Dentre as principais formas de com-

bate a possível disseminação do vírus nessa localidade, ressalta-se as recomendações feitas pela OMS (Organização Mundial de Saúde), como a utilização de álcool para desinfecção de superfícies e produtos e a manutenção do distanciamento social, limitando o número de indivíduos presentes em um mesmo local e mantendo uma distância mínima entre estes [2].

Matrajt et al. aponta a necessidade de manter uma distância mínima entre as pessoas e limitar o número de pessoas presentes em um mesmo local para a redução dos casos do novo coronavírus [3]. Organizações como a OMS recomendam um distanciamento mínimo de um metro entre cada indivíduo, além de evitar aglomerações sociais [2].

A eficiência do álcool comum no combate ao coronavírus já foi comprovada por meio de diversos estudos. Desse modo, é de extrema relevância que o estabelecimento possuam sempre uma fonte de álcool em gel para a higienização das mãos e produtos comprados, além de qualquer outro objeto ou superfície que possa vir a conter o vírus [4]. Tendo em vista essa problemática, a tecnologia pode vir a ser de extremo auxílio a fim de manter as boas práticas de prevenção do COVID-19. Ao utilizar-se de formas automatizadas para garantir que as recomendações de segurança sejam seguidas, reduz-se a quantidade de trabalhadores em um local, e, portanto, de indivíduos neste, reduzindo o número de pessoas em risco, além de aliviar aglomerações [5].

Logo, este trabalho tem como finalidade a criação de um ambiente seguro e automatizado, a fim de restringir ao máximo a contaminação pelo vírus, contextualizado em um supermercado. O projeto foi pensado para auxiliar na resolução de dois problemas comuns: falta de higienização e manutenção do distanciamento social. Desse modo, foram desenvolvidos três dispositivos utilizando a tecnologia do Arduino:

- Um controlador de fluxo a partir da contabilização do número de pessoas que entram e saem do ambiente do supermercado;
- Um distribuidor automático de álcool em gel que evita o contato direto;
- Um sistema de controle do distanciamento social nas filas dentro do supermercado.

2. Fundamentação teórica

Com o propósito de redução na disseminação do SARS-CoV-2, os supermercados devem atentar-se quanto ao distanciamento entre os clientes dentro dos estabelecimentos. De acordo com análises de circulação nas lojas, a Associação Brasileira de Supermercados recomenda a permanência de uma pessoa por 12 m² de área de vendas [6] (Figura 1).

Área (m²)	Quantidade de clientes (un.)
500	42
1.000	83
2.000	167
3.000	250
5.000	417
7.000	583
10.000	833

Figura 1. Número de clientes por área total de vendas.
Fonte: ABRAS, 2020.

Juntamente com a redução do número de indivíduos em um local, é também necessário a manutenção de uma distância mínima individual entre aqueles ainda presentes. Sabe-se que o vírus em questão possui alto poder de propagação, tornando-se, portanto, essencial uma distanciação de pelo menos um metro entre cada pessoa [7].

Posteriormente ao controle de aglomerações, está a utilização do álcool 70% para a desinfecção de superfícies potencialmente portadoras do vírus. Pelo fato do SARS-CoV-2 ser uma célula com uma capa lipoproteica, o álcool consegue dissolver a gordura e penetrar o interior da célula, onde este desnatura as proteínas presentes, resultando na morte do vírus [8].

A eficácia dessas medidas é comprovada e recomendadas por órgãos especialistas no assunto. Mesmo que não seja possível extinguir o vírus completamente, tais medidas possuem a capacidade de "achatar a curva" do coronavírus, aliviando a tensão imposta no sistema de saúde e permitindo um melhor tratamento dos pacientes aflitos (Figura 2)

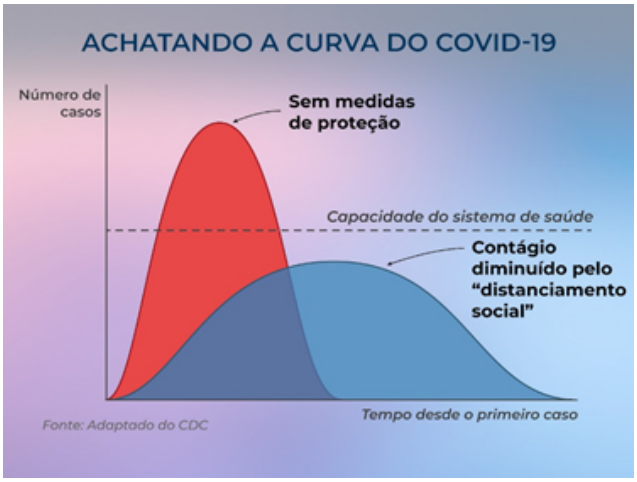

Figura 2. Achatamento da curva de contágio.
Fonte: Unisul,2020

3. Desenvolvimento e Simulação computacional

Com base na fundamentação teórica, foi desenvolvido ao longo desse projeto três tecnologias com o propósito de automatizar e facilitar o seguimento dessas recomendações.

Primeiramente, desenvolveu-se um contador de pessoas para a entrada de estabelecimentos, com o objetivo de regular o número de pessoas circulando em um local. Inicialmente colocado a uma altura estratégica de 95 cm para registrar indivíduos de diferentes alturas, este regula a entrada e saída de pessoas, fazendo que assim seja respeitado o limite de uma pessoa para cada 12 m² de área, conforme as recomendações da ABRAS [6].

Desenvolveu-se, também, um "dispenser" de álcool automático, que evita o contato direto com o consumidor e proporciona a quantidade correta do produto a ser utilizada para a desinfecção própria do indivíduo.

Por fim, foi desenvolvido um protótipo para a manutenção da distância mínima de um metro entre indivíduos enquanto estes encontram-se em uma fila, fazendo uso de sensores ultrassônicos e avisos sonoros e visuais.

3.1. Controle de Entrada e Saída

Conhecer o número de indivíduos presentes no interior de um recinto é de extrema relevância para evitar a formação de aglomerações de pessoas, como pode ocorrer em supermercados. A partir dessa informação, os funcionários podem ter um maior controle da entrada de clientes e seguir as normas do protocolo de lotação máxima de acordo com a área total de vendas.

Em razão disso, uma solução tecnológica simples seria a implementação de um dispositivo de contagem que utiliza o Arduino, sensores de luz e lasers, que são capazes de detectar a presença e a direção do movimento de uma pessoa, possibilitando a atualização do número total de pessoas presentes. Tanto a lotação atual quanto a capacidade máxima seriam apresentadas por meio de um display de LCD, que poderia ser visto tanto pelos funcionários quanto pelos clientes. Ademais, o dispositivo também alertaria as pessoas presentes, a partir de um LED vermelho e do display, caso o limite máximo pré-estabelecido for ultrapassado.

3.1.1 Dispositivo - Controlador de fluxo

O uso conjunto de dois desses dispositivos nas duas entradas de um corredor, com uma simples mudança no software e hardware para implementar uma comunicação sem fio entre os aparelhos, possibilitaria a contagem simultânea de pessoas presentes dentro de um corredor. Desse modo, sempre que o contador de um dos dispositivos fosse modificado, o outro também seria atualizado conformemente.

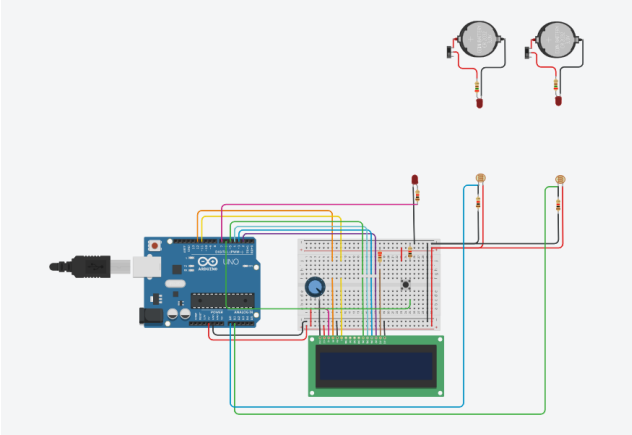

Figura 3. Esquematização do circuito para contagem e saída, pelo tinkercad. Fonte: Autores, 2020

Tabela 1. Componentes Controlador de Fluxo, Fonte: Autores, 2020.

Quantidade	Componente
1	Arduino Uno R3
1	LCD 16 x 2
1	Resistor de 220 Ω
1	Botão
1	Potenciômetro
2	Fotorresistor (LDR)
2	Diodo Laser 5mW
1	Vermelho LED
4	Resistor de 1kΩ

3.1.2 Lógica e código

O programa, apresentado no apêndice X, pode ser dividido em três partes principais: A lógica de contagem, o funcionamento após ultrapassar a capacidade máxima e a função de calibragem.

3.1.3 Lógica para a contagem de pessoas

Neste projeto, foram utilizados 2 lasers e 2 fotorresistores (LDR) para contar o número de entradas e saídas baseado na ordem de interrupção dos lasers. Para tal, foi criada a variável "order", uma string que indica a ordem de interceptação, possibilitando a atualização do número de pessoas presentes no recinto. Caso "order" seja igual a "1-2", significa que uma pessoa entrou. Se for igual a "2-1", alguém saiu. As variáveis "triggerValue1" e "triggerValue2" representam os valores máximos de luminosidade que indicam que o laser não está atingindo o sensor. Estes são os parâmetros para identificar se algum dos lasers foi interrompido. Ambos podem ser modificados pela função "calibrate()" para que o dispositivo funcione ade-

quadamente em diferentes ambientes. Quando o primeiro laser for interceptado, a leitura do sensor será igual ou menor que o valor de “triggerValue1”, assim, o programa escreve “1” na variável “order”, caso já não esteja escrito. O mesmo ocorre para o segundo laser. Caso ambos os laser sejam interrompidos, o programa adiciona “-” a “order”. Assim, o sistema reconhece como uma entrada ou uma saída apenas nos casos “1-2” e “2-1”, sendo o primeiro uma entrada e o segundo uma saída. Se “order tiver dois caracteres e o segundo não é “-”, a variável é apagada. A decisão de utilizar o “-” como um indicador de que ambos os lasers foram interceptados foi tomada para prevenir algumas inconsistências da lógica com apenas “1” e “2”. Um exemplo seria: uma pessoa interrompe o primeiro laser, volta e outra pessoa interrompe o segundo. Neste caso, o programa iria contabilizar um entrada errônea. Manter obrigatório que que ambos os lasers sejam interrompidos simultâneamente para contabilizar uma entrada ou saída resolve este e outros possíveis erros.

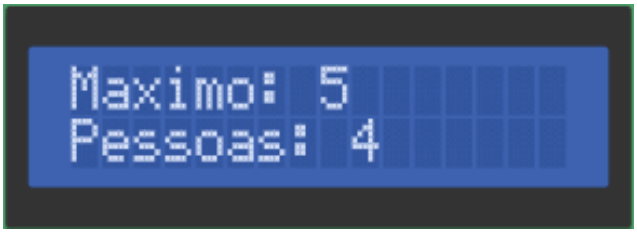

Figura 4. Mensagem padrão do contador.
Fonte: Autores, 2020

3.1.4 Capacidade Máxima ultrapassada

A execução da função loop() sempre verifica se o número de pessoas do contador alterou durante esse ciclo do loop e compara se a lotação é maior que a capacidade máxima pré-definida. Se o limite máximo foi ultrapassado nesse ciclo, o programa apresenta no LCD uma mensagem singular (Figura 5), com um segundo de duração, demonstrando que o limite foi atingido. Ademais, a função blinkLED() é chamada, que inicia uma repetição no estado do LED vermelho, fazendo com que ele pisque, como pode ser observado na figura 7, até que o limite máximo seja respeitado novamente. Depois, uma outra mensagem é mostrada (Figura 6) que indica, além do fato de que a capacidade máxima foi ultrapassada, a quantidade de pessoas no recinto. Finalmente, se o contador for atualizado e verificar que a quantidade de pessoas voltou a ser menor ou igual ao limite máximo, então o programa retorna a apresentar a mensagem padrão (Figura 4).

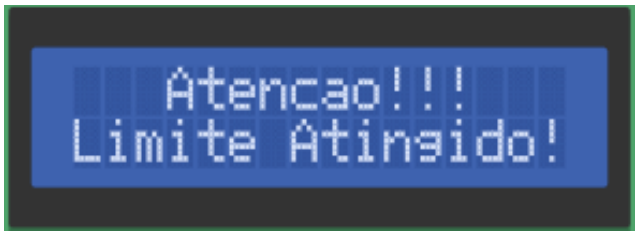

Figura 5. Mensagem mostrada ao ultrapassar o limite máximo.
Fonte: Autores, 2020

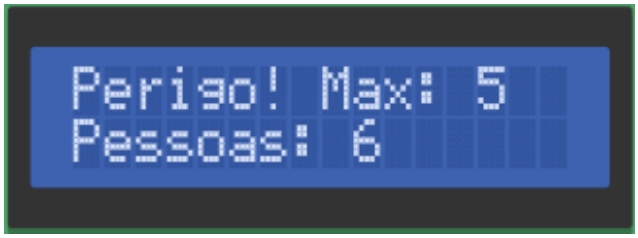

Figura 6. Mensagem padrão quando a lotação é maior que o limite pré-definido.
Fonte: Autores, 2020

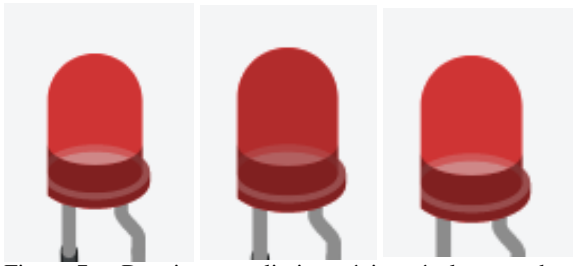

Figura 7. Depois que o limite máximo é ultrapassado o LED vermelho começa a piscar Fonte: Autores, 2020

3.1.5 Função de Calibragem dos Sensores

Diferentes ambientes fazem com que os sensores captem uma variedade distinta de luminosidade, causando um mal funcionamento do aparelho. Desse modo, a função de calibragem foi adicionada ao programa para ajustar os valores de gatilho, que são utilizados para determinar se uma pessoa está interceptando os lasers. Durante a execução do loop(), no início de cada ciclo, o programa sempre executa a função buttonDetect(), o qual detecta se o botão de calibragem foi apertado ou não. Caso, tenha sido pressionado, a variável “numClicks” é atualizada para 1 e prepara o programa para detectar pela segunda vez, caso o botão seja pressionado novamente. Uma importante observação é que, devido ao fato de que o botão não precisa ser solto, segurar o botão por um breve intervalo é interpretado pelo programa do mesmo modo que pressionar duas vezes em sequência. Ambas situações acionam o programa para ele chamar a função calibrate().

Essa segunda função inicia, então, o processo de calibragem, que irá apresentar continuamente na tela do LCD instruções para o usuário corretamente calibrar os sensores. Dessas mensagens, algumas mostram também um temporizador de três segundos, que pode ser alterado, no canto inferior direito da tela, disponibilizando, assim, tempo suficiente para o leitor. Durante o procedimento, o programa inteiro vai ser pausado duas vezes e apresentar as mensagens na Figura 8, aguardando o usuário pressionar o botão duas vezes para que os valores, da luminosidade captada do ambiente quando os sensores não estão sendo atingidos pelos lasers e quando estão sendo atingidos, sejam lidos pelo programa após o recinto estar corretamente arrumado. Esse valores lidos são, então, mostrados na tela do LCD (Figura 9). Depois disso, a função calcula a média aritmética entre os valores de luminosidade lidos e estabelece esses valores como os valores de gatilho para cada sensor, apresentando os números calculados na tela (Figura 10). Finalmente, a função verifica se o valor lido de algum dos sensores quando esse estava sendo interceptado pelo laser for maior que o valor lido quando não estava sendo interceptado. Essa condição indica que a calibragem ocorreu corretamente e apresentará uma mensagem informando isso (Figura 11). Caso a condição não tenha sido satisfeita, a função mostra uma mensagem de falha na calibragem (Figura 11) e chama ela mesma, iniciando novamente o processo de calibragem.

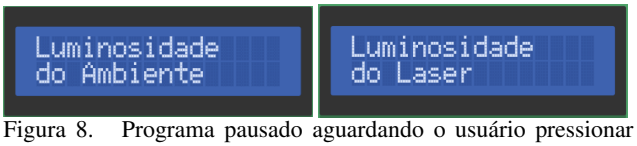

Figura 8. Programa pausado aguardando o usuário pressionar o botão para os sensores captarem os valores de luminosidade. Fonte: Autores, 2020.

Figura 9. Informando os valores lidos pelos sensores. Fonte: Autores, 2020.

3.1.6 Instalação e Simulação

O dispositivo foi pensado de modo que consiga corretamente manter a contagem para medidas de diferentes pessoas, abrangendo desde crianças até idosos. Para isso, recomenda-se que o dispositivo seja instalado a uma altura de 95 cm, visando abranger múltiplas alturas, e os lasers devem ser posicionados de modo que fiquem na mesma altura, seus feixes sejam paralelos e tenham um distância de 7 cm entre si.

Abaixo, será apresentado o funcionamento da lógica de contagem utilizada pelo dispositivo:

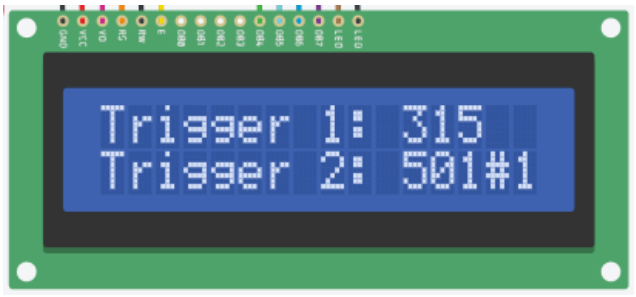

Figura 10. Valores calculados de gatilho para os sensores. Fonte: Autores, 2020

Figura 11. Resultados possíveis ao fim da função. Fonte: Autores, 2020.

Entrada:

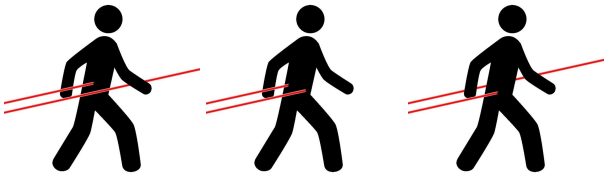
Figura 12. Representação de uma pessoa entrando. Fonte: Autores, 2020.

Saida:


Figura 13. Representação de uma pessoa saindo. Fonte: Autores, 2020.

3.1.7 Problemas, Erros e Hipóteses

O programa foi projetado para lidar com possíveis erros e com as vulnerabilidades do hardware de forma robusta, porém seriam necessários testes com pessoas reais para reduzir ainda mais as imprecisões de contagem. Dentre as medidas tomadas para diminuir a ocorrência de problemas estão, a criação de um indicador de que ambos os lasers foram interceptados, a manutenção de uma distância pequena entre si e adição uma função de calibragem para

diferentes iluminações. Algumas das limitações do dispositivo são:

- Requer que uma pessoa passe por vez.
- A luz ambiente pode interferir na leitura dos sensores de luz, dificultando a identificação da presença do laser no sensor.
- Crianças no colo de outra pessoa são detectadas como uma única pessoa

3.2. Distribuidor de álcool

Para o problema com higienização e constante, existem múltiplas soluções, a de maneira mais prática e barata seria a criação de um distribuidor de álcool. Foi montado um protótipo de dispensador automatizado para a distribuição de álcool-70 (recomendado para desinfecção bacteriana). Tais dispensadores ficariam estrategicamente localizados nas posições de entrada, saída, comes e bebes, freezers, armários e caixa, enfim, locais de alto potencial de infecção. Com sua utilização poderia ser reduzido altamente o vírus que foi adquirido ao se pegar em corrimões, produtos e/ou ao entrar em contato físico com outra pessoa.

3.2.1 Projeto distribuidor de álcool

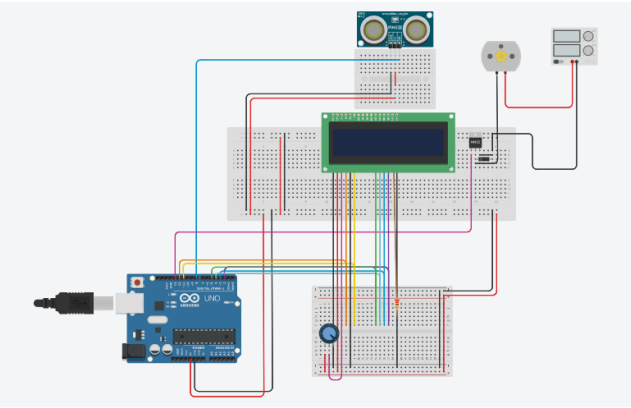

Figura 14. Esquematização do circuito-projeto para distribuição de álcool. Fonte: Autores, 2020.

3.2.2 Perspectiva do usuário:

Hipotetizando uma pessoa ao entrar, ela veria o dispositivo logo na entrada (Figura 15) ou lateralmente (Figura 16) de maneira que ela veria a uma mensagem e logo utilizaria o dispositivo (Figura 16).

Ao aproximar as mãos, o dispositivo esguicharia álcool na mão do usuário por cerca de 5 segundos, depois exibiria a mensagem “Retire suas mãos” durante 2 segundos, já tendo

Tabela 2. Lista de materiais do circuito-projeto para distribuição de álcool. Fonte: Autores, 2020.

Quantidade	Componente
1	Arduino Uno R3
1	Fonte de energia de 3V e 5A
1	Motor CC
1	LCD 16 x 2
1	Sensor de distância ultrassônico
1	Transistor nMOS (MOSFET)
1	Diodo
1	Potenciômetro
1	Resistor de 220 Ω

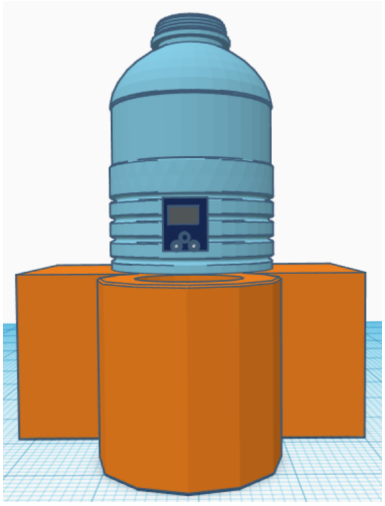

Figura 15. Projeção 3D frontal do projeto para distribuição de álcool. Fontes: Autores, 2020.

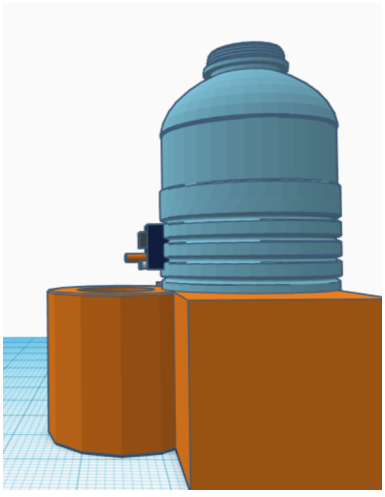

Figura 16. Projeção 3D Lateral do projeto para distribuição de álcool. Fonte: Autores, 2020.

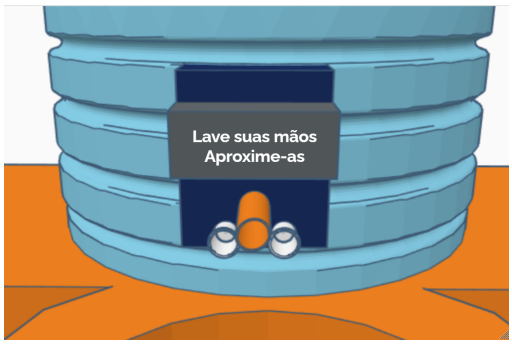

Figura 17. Ênfase frontal no dispositivo. Fontes: Autores, 2020.

parado o esguicho, reiniciaria o processo voltando a exibir "Lave suas mãos, aproxime-as".

3.2.3 Lógica do sistema:

Liga-se o projeto em uma fonte direto na tomada (representação pela figura 14). Após ser ligado (Figura 18), ele se inicia em um estado suspenso, apenas esperando alguma mudança no estado de seu sensor de distância. Nesse estado inicial ele apresenta a mensagem "Lave suas mãos, aproxime-se" (Figura 17 e Figura 19) referência a que proximidade irá ativar o sistema de distribuição de álcool, ao se aproximar por menos de 31 cm o sistema começa a distribuição por 5000ms de álcool, após isso ele exibirá uma mensagem de cerca de 2000ms dizendo para retirar a mão e com isso o sistema reiniciará para o estado suspenso

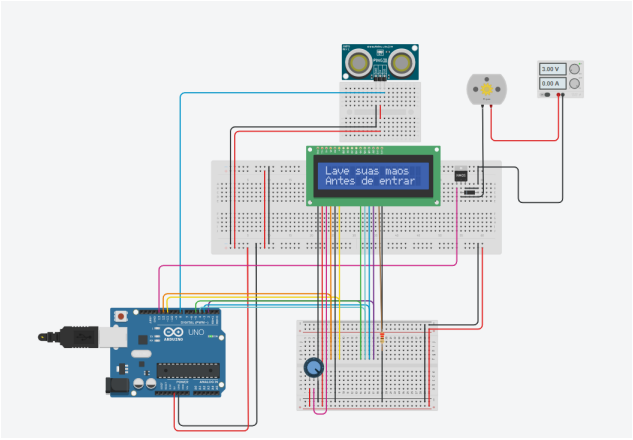

Figura 18. Produto ao ligar. Fonte: Autores, 2020.

3.2.4 Explicação do código e hardware

Existe a declaração de variáveis no começo do escopo do código (Figura 20), determinação das portas da placa Arduino do sensor Ultrassônico e do Painel LCD, além da

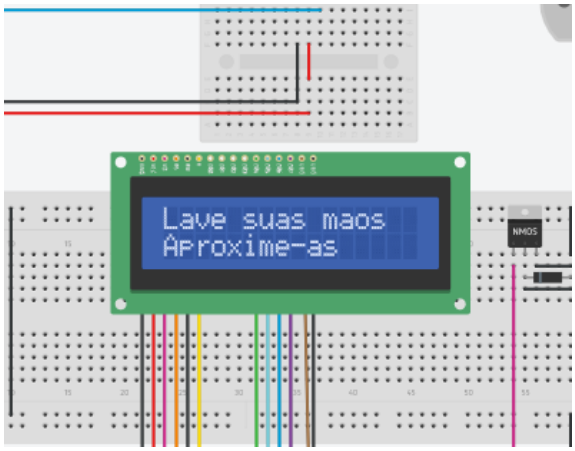

Figura 19. Primeira mensagem apresentada. Fonte: Autores, 2020.

criação da função que recebe os valores de entrada e saída do sensor ultrassônico e fornece uma constante ultrassônica

Foi utilizado o sensor ultrassônico Parallax Ping, que é capaz de determinar o distanciamento de objetos e pessoas em até 3 metros. No código foi determinada a distância de até para a proximidade e funcionamento como sendo de 30 cm.

```
#include <LiquidCrystal.h>

int cm = 0;
int motor = 13;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

long readUltrasonicDistance(int triggerPin, int echoPin)
{
  pinMode(triggerPin, OUTPUT);
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);
  return pulseIn(echoPin, HIGH);

}
```

Figura 20. Início do escopo do código. Fonte: Autores, 2020.

Na função de configuração (Figura 21) é definido que a leitura usb é de 9600 ciclos bps. Que na porta 13 (que foi definida no começo do código) é um motor, com frequência de saída. Também é inicializado o painel lcd com 16 colunas e 2 linhas.

Foi utilizado o painel LCD por sua facilidade de uso e para instruir ao usuário durante sua utilização.

Como não existe uma bomba d'água no TinkerCad (plataforma usada para projetar o sistema) foi utilizado um

motor cc para representar tal.

```
void setup()
{
  Serial.begin(9600);
  pinMode(motor, OUTPUT);
  lcd.begin(16, 2);
}
```

Figura 21. Função de configuração da placa. Fonte: Autores, 2020.

Na parte do loop de funcionamento (Figura 22), primeiro se converte a constante ultrassônica para cm a partir da multiplicação por 0,01723 (valor já disponibilizado na biblioteca). Após isso é comparado o valor em cm 30, se for, ele mostra no LCD que está jorrando álcool, além de ligar o motor como representação, após 5000ms (5s) o processo acaba, o motor se desliga e é pedido para o usuário retirar suas mãos. Se a distância for maior que 30 após 2 segundos ou até mesmo no começo do loop, o motor fica desligado e apenas exibe que o próximo usuário preferencialmente deve se aproximar.

3.2.5 Explicação do circuito elétrico:

Conforme a Figura 23 e a Figura 24, é possível observar todo o sistema elétrico, mas passando por partes específicas dele existem suas peculiaridades, como o sistema, placa Arduino está sendo alimentada de forma usb, não seria suficiente para o motor girar, logo o motor está sendo alimentado por uma fonte de 3v e 5a, capacidade do motor.

Todo o sistema está ligado na porta 13 digital do Arduino, que está ligado a MOSFET (transistor) e um diodo para transformar a corrente alternada em contínua e controlar a abertura e fechamento do fluxo de energia (0 e 1). Também poderia ser utilizado uma ponte H.

Nas portas 12, 11, 5, 4, 3, 2 do arduino digital estão as ligações do LCD (Figura 24), tirando a alimentação. É utilizado um potenciômetro de 100k ohm para controlar o fluxo de energia, logo do LCD.

Por último existe o sensor Ultrassônico, com alimentação de 5v e controle na porta 8 digital.

3.3. Sistema de distanciamento em filas

Sabendo que a taxa de transmissão do Coronavírus é diretamente proporcional à proximidade entre o indivíduo transmissor e o receptor, foi-se desenvolvido um sistema a fim de promover o distanciamento social entre os consumidores enquanto estes esperam em uma fila para serem atendidos. Tal sistema faz uso de avisos visuais e sonoros para que a distância entre cada indivíduo na fila seja de, pelo menos, um metro, conforme recomendado pela OMS [2]. Ao utilizar-se de um dispositivo como este, seria possível evitar contaminações por proximidade elevada, diminuindo a taxa de contágio dentro do estabelecimento e aumentando a segurança dos clientes.

```
void loop()
{

  cm = 0.01723 * readUltrasonicDistance(8, 8);
  delay(1000);
  Serial.print(cm);
  Serial.println("cm");

  if(cm<=30){
        lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Lavando...");
    digitalWrite(motor, HIGH);
    lcd.setCursor(0,1);
    lcd.print("Use mascara! :)");
    delay(5000);
    lcd.clear();
    lcd.print("Retire suas maos :)");
    digitalWrite(motor, LOW);
        delay(2000);
        }
  else{
    digitalWrite(motor, LOW);
    lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Lave suas maos");
        lcd.setCursor(0,1);
        lcd.print("Aproxime-as");
    delay(500);
  }
}
```

Figura 22. Parte do loop de funcionamento. Fonte: Autores, 2020.

3.3.1 Projeto controlador de distanciamento social

3.3.2 Perspectiva do usuário

Supondo um indivíduo na fila de um supermercado qualquer, este deveria manter um distanciamento mínimo de um metro de distância de qualquer outro indivíduo no recinto. Para auxiliar na manutenção dessa distância, o usuário, enquanto na fila, possuiria um display de LCD em sua frente,

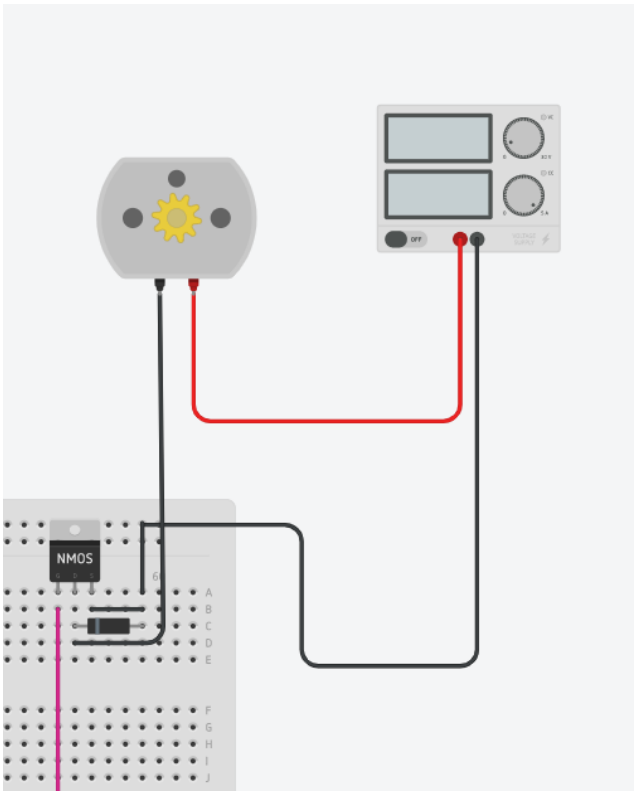

Figura 23. Demonstração do motor e sua fonte. Fonte: Autores.

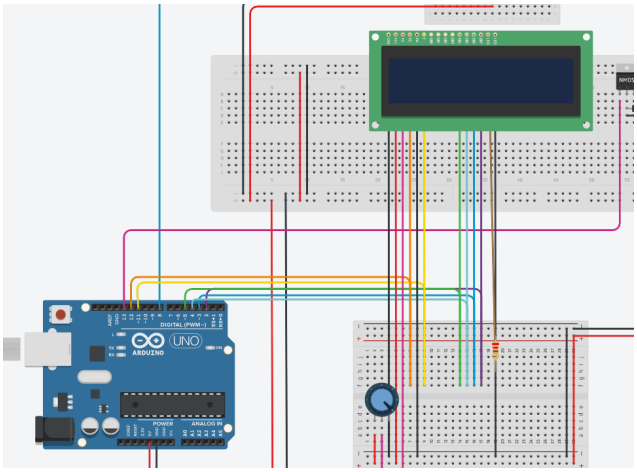

Figura 24. LCD e potenciômetro. Autores: 2020.

lhe informando sobre a distância entre este e o local ocupado à sua frente. Além disso, caso o usuário cruze a distância mínima de um metro, soará um buzzer até que volte o distanciamento recomendado, juntamente com um LED RGB que mudará de cor conforme a distância mantida.

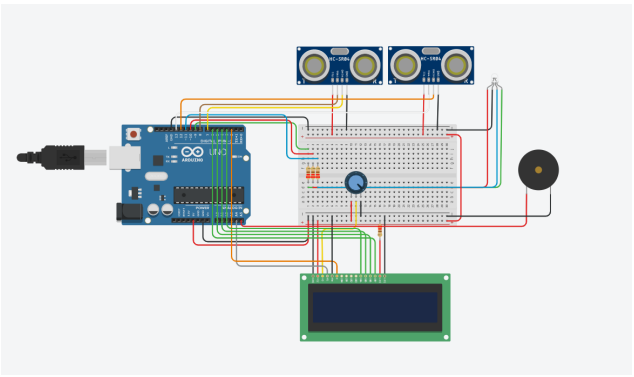

Figura 25. Esquematização do circuito-chave do sistema de distanciamento em filas. Fonte: Autores, 2020.

Tabela 3. Componentes utilizados no protótipo do sistema de distanciamento em filas. Fonte: Autores, 2020.

Quantidade	Componente
1	Arduino Uno R3
1	LED RGB
1	LCD 16 x 2
4	Resistor de 220 Ω
2	Sensor de distância ultrassônico
1	Potenciômetro
1	Piezo

3.3.3 Lógica do sistema

O sistema possui quatro estados de funcionamento principais: o primeiro sendo quando não há ninguém no local a frente do usuário na fila, onde o LED e o Buzzer ficarão desligados e o display LCD mostrará as palavras “Próximo” (Figura 26); o segundo quando houver um indivíduo no local a frente do usuário na fila, a pelo menos um metro e meio de distância, ligando o LED na cor azul e desligando o display LCD (Figura 27); o terceiro no caso em que a distância mantida entre os dois indivíduos estiver entre um metro e um metro e trinta centímetros, onde o LED torna-se amarelo, o buzzer apita levemente e o display mostra uma mensagem pedindo “Afaste-se, por favor” (Figura 28); e por fim o quarto, quando a distância é inferior à um metro, onde o LED torna-se vermelho e o buzzer solta um forte sinal sonoro (Figura 29).

3.3.4 Explicação do Código/Hardware

O código inicia-se com a inicialização das variáveis (Figura 30), definindo os pinos a serem utilizados para a placa LCD, para o LED RGB e para o Buzzer. Logo em seguida faz-se o setup destes pinos, definindo-os como OUTPUT, no caso do LED, dos pinos de transmissão do sensor ultrassônico e do Buzzer, e INPUT no caso dos pinos

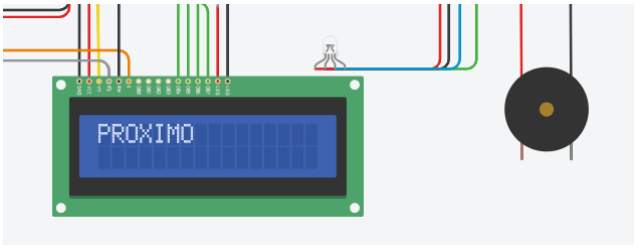

Figura 26. Primeiro estado do sistema. Fonte: Autores, 2020.

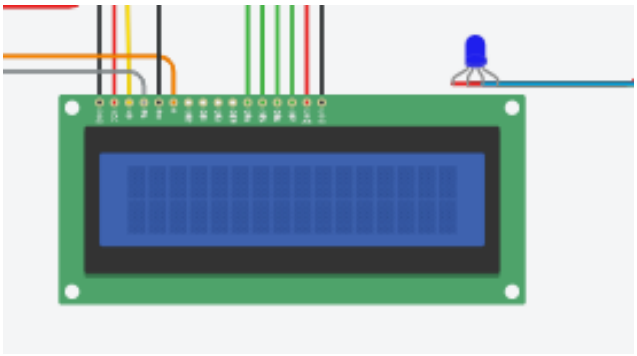

Figura 27. Segundo estado do sistema. Autores, 2020.

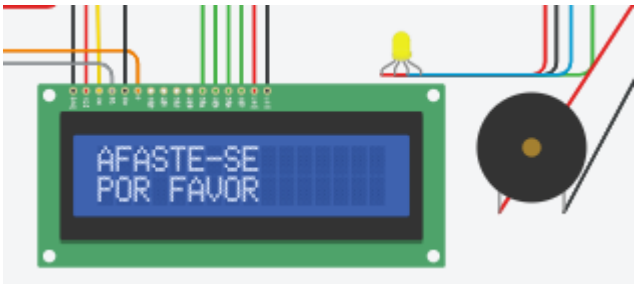

Figura 28. Terceiro estado do sistema. Autores, 2020.

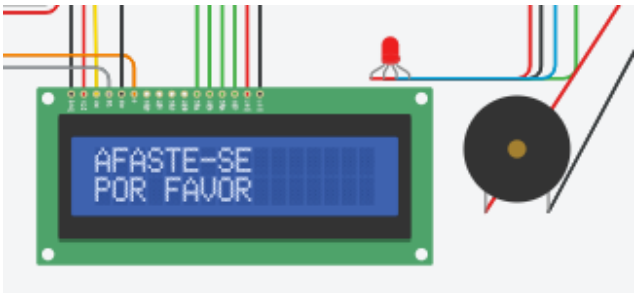

Figura 29. Quarto estado do sistema. Autores, 2020.

de recepção do sensor ultrassônico. Além disso, inicia-se a tela LCD como uma tela 16x2 (Figura 31).

Logo em seguida, tem-se o loop principal, onde ocorre boa parte do programa. Primeiramente, definem-se algu-

```
//Lucas Ramson
#include <LiquidCrystal.h>

float cm,duracao, alguem;

//definindo pinos da placa LCD LiquidCrystal lcd(<pino RS>,
//<pino enable>,<pino D4>, <pino D5>, <pino D6>, <pino D7>)
LiquidCrystal lcd(A4, 2, 3, 4, 5, 6);

//Sensores
byte pinoTrans1 = 8;  //pino trig 1
byte pinoRec1 = 7; //pino echo 1
byte pinoTrans2 = 13; //pino trig 2
byte pinoRec2 = 12; //pino echo 2

//LED
int pinoR = 10;
int pinoG = 9;
int pinoB = 11;

//Buzzer
int buzzer = A5;
```

Figura 30. Inicialização das variáveis. Autores, 2020.

```
void setup()
{
  //Display LCD / Colunas e Linhas
  lcd.begin(16,2);

  //LED
  pinMode(pinoR, OUTPUT);
  pinMode(pinoG, OUTPUT);
  pinMode(pinoB, OUTPUT);

  //SENSOR
  pinMode(pinoTrans1, OUTPUT);
  pinMode(pinoTrans2, OUTPUT);
  pinMode(pinoRec1, INPUT);
  pinMode(pinoRec2, INPUT);

  //BUZZER
  pinMode(buzzer, OUTPUT);

  //iniciar a porta serial
  Serial.begin(9600);


}
```

Figura 31. Setup das variáveis. Autores, 2020

mas variáveis e limpa-se a tela LCD. Logo em seguida, é feita a primeira checagem: se há alguém na posição à frente do usuário na fila. Caso positivo, segue-se a checagem de distância, da menor para a maior. Inicialmente, no caso de uma distância inferior a 100 cm, ou seja, 1 metro (Figura 32). Faz-se então, a checagem para uma distância superior a 100 cm e inferior a 150 cm, e por fim, distâncias superiores a 150 cm (Figura 33). Caso a primeira condição não seja cumprida, observa-se que não há nenhum indivíduo na

outra posição da fila (Figura 34). Com base no caso observado, tem-se a reação esperada, conforme explicado na parte 3.3.1.2 deste artigo.

```
void loop()
{
  delay(1000);
  lcd.clear();
  delay(500);
  cm = distancia(); //distancia em cm
  alguem = distancia1();
  if (alguem > 0){ //checa se há alguém a frente na fila
     //limpa a tela
    //acender o LED com base na distancia, mudando de cor.
    //distancia de 1m (distanciamento social! :))
    if(cm >0 && cm<=100) { //vermelho caso muito perto
      analogWrite(pinoR, 255);
      analogWrite(pinoG, 0);
      analogWrite(pinoB, 0);
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("AFASTE-SE");
      lcd.setCursor(0,1);
      lcd.print("POR FAVOR");
      Serial.print("AFASTE-SE POR FAVOR");
      tone(buzzer,250);
      delay(500);
```

Figura 32. Início do loop. Autores, 2020.

```
    }
    else if (cm >100 && cm <=150){ //amarelo caso se aproximando
      analogWrite(pinoR, 255);
      analogWrite(pinoG, 255);
      analogWrite(pinoB, 0);
      lcd.clear();
      lcd.print("AFASTE-SE");
      lcd.setCursor(0,1);
      lcd.print("POR FAVOR");
      Serial.print("AFASTE-SE POR FAVOR");
      tone(buzzer,200);
      delay(500);

    }
    else if (cm>150){ //azul caso longe
      analogWrite(pinoR, 0);
      analogWrite(pinoG, 0);
      analogWrite(pinoB, 255);
      lcd.clear();
      noTone(buzzer);
      delay(500);
```

Figura 33. Outras checagens de distância. Autores, 2020.

Finalmente, definem-se algumas funções para facilitar o código. Define-se a função distancia() (Figura 35) a fim de calcular a distância do primeiro sensor ultrassônico em centímetros. Para tal, emite-se um sinal sonoro e calcula-se o tempo necessário para este ir e voltar até o sensor. Utiliza-se da mesma lógica para a função distancia1(), apenas mudando para o segundo sensor ultrassônico.

4. Conclusão

Pode ser observado, a tecnologia e automação têm sido aliados formidáveis para o combate ao vírus, em razão disso, a realização de um projeto que, na atual fase da pandemia, consegue ajudar no combate e na prevenção do vírus em ambientes essenciais é vital. Sim, é importante que os estabelecimentos sigam as normas para colaborar com o

```
    }
    else{
      lcd.clear();
      analogWrite(pinoR, 0);
      analogWrite(pinoG, 0);
      analogWrite(pinoB, 0);
      lcd.setCursor(0,0);
      lcd.print("PROXIMO");
      Serial.print("PROXIMO");
      delay(500);
      noTone(buzzer);
    }
  }
```

Figura 34. No caso de não existir algum indivíduo à frente do usuário. Autores, 2020.

```
}
//funcao para calcular a distancia:
float distancia() {
  //limpar o pino/aguardar
  digitalWrite(pinoTrans2, LOW);
  delayMicroseconds(1);
  //pulso do ultrassom
  digitalWrite(pinoTrans2, HIGH);
  delayMicroseconds(10);
  digitalWrite(pinoTrans2, LOW);
  //calcula a duracao em ms para ida e volta do pulso
  duracao = pulseIn(pinoRec2, HIGH);
  float  calcDistancia = (duracao/2) * 0.0343;
  if (calcDistancia>=331){
    calcDistancia=0;
  }
  delay(500);
  return calcDistancia;
}
```

Figura 35. Definição da função distancia(). Autores, 2020.

achatamento da curva do SARS-CoV-2, e que todos façam sua parte utilizando todos os meios disponibilizados para a prevenção e disseminação do vírus e, ambientes. Mas aliando-se a tecnologia, pode-se ter uma redução drástica de casos.

Deste modo, o projeto desenvolvido tem intenção de automatizar tarefas colaborando como frente à pandemia. Não expondo nenhum ser humano ao risco que é o COVID-19. O projeto teve seus três pilares constituídos através de pesquisas bibliográficas, necessidade social e relevância/impacto, Toda a pesquisa bibliográfica foi essencial para o desenvolvimento, informações como a distância social adequada, fluxo de pessoas e eficiência do álcool foram primordiais para a construção dos projetos.

Os três protótipos do projeto têm como objetivo a redução e prevenção da propagação do COVID-19. Auxiliam no controle de aglomerações a partir da contabilização de pessoas ao adentrar os ambientes e distanciamento entre os clientes nas filas dos caixas, com a disponibilização de álcool, é possível reduzir a infecção por locais e objetos

infectados.

Como perspectiva e planos para o futuro, pelo projeto ter sido desenvolvido por estreantes na área e para a matéria de Introdução a Engenharia Mecatrônica, procura-se melhorar os projetos, procurar parcerias, além de aumentar a eficiência do projeto. Seria importante desenvolver a automação e precaução em outras partes do supermercado, bem como a desinfecção constante do ambiente, além da automação completa de caixas.

Referências bibliográficas

[1] A. Rauf, "How is technology helping to fight coronavirus?," 05 2020.

[2] W. H. Organization *et al.*, "Overview of public health and social measures in the context of covid-19: interim guidance, 18 may 2020," tech. rep., World Health Organization, 2020.

[3] L. Matrajt and T. Leung, "Evaluating the effectiveness of social distancing interventions to delay or flatten the epidemic curve of coronavirus disease," *Emerging infectious diseases*, vol. 26, no. 8, p. 1740, 2020.

[4] R. A. Leslie, S. S. Zhou, and D. R. Macinga, "Inactivation of sars-cov-2 by commercially available alcohol-based hand sanitizers," *American Journal of Infection Control*, 2020.

[5] X. Geng, F. Gerges, G. G. Katul, E. Bou-Zeid, H. Nassif, and M. C. Boufadel, "Population agglomeration is a harbinger of the spatial complexity of covid-19," *Chemical Engineering Journal*, p. 127702, 2020.

[6] A. Brasil, *protocolo para prevenção do coronavírus (covid-19) nos supermercados*$_2$020. 5 *ed.*, 2020.

[7] H. Stevens, "Why outbreaks like coronavirus spread exponentially, and how to "flatten the curve"," *Washington Post*, vol. 14, 2020.

[8] R. A. Leslie, S. S. Zhou, and D. R. Macinga, "Inactivation of sars-cov-2 by commercially available alcohol-based hand sanitizers," *American Journal of Infection Control*, 2020.

Apêndice

A. Código Controle de Entrada e Saída

```
#include <LiquidCrystal.h>
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// BlinkLed
int ledPin = 7; // Led no pino 7
int ledState = 0; // Esse led comeca a
    piscar quando ultrapassa a capacidade
    maxima
int period = 500; // tempo entre o led
    ligado e desligado
unsigned long long ledTime = 0; //
    variavel utilizado para a funcao
    blinkLed()

// Sensores
int ldrPin1 = A0; // LDR1 no pino
    analogico 8
int ldrPin2 = A1;
int ldrValue1 = 0; // Valor que vai ser
    lido do LDR1
int ldrValue2 = 0; // Valor que vai ser
    lido do LDR2
int triggerValue1 = 350; // Valor para
    disparar o sensor -> Valores maiores
    disparam o sensor
int triggerValue2 = 350; // Valor impreciso

// Calibragem
int buttonState = 0;
int buttonPin = 6;
unsigned long timePressed = 0;
unsigned long timePressedLimit = 0;
int numClicks = 0;
int ambientLuminosity1 = 0;
int ambientLuminosity2 = 0;
int laserLuminosity1 = 0;
int laserLuminosity2 = 0;

// Logica de Entrada
int people = 0; // Quantidade de pessoas
    dentro
int maxPeople = 5; // Capacidade Maxima de
    pessoas
bool hasNumPersonChanged = false;
bool hasReachedMax = false;
bool hadPreviousReachedMax = false;
unsigned long long
    firstReachedMax_LCDPrintTime = 0;
bool firstReachedMax = false;
String order = ""; // string contendo
    ordem de ativacao
String lastChar = ""; // ultimo caractere
    da ordem para comparacao

///Prototipos das Funces
void printLcd(); // imprime o padrao do lcd
void reachedMaxPrint(); // dispara quando
    o limite de pessoas for atingido
void blinkLed();
void readSensors(); // ler os valores dos
    2 sensores LDR
void calibrate(); // falta implementar!
    vai automatizar o trigger
void buttonDetect(); // verifica se o
    usuario esta pressionando o botao
void printTimer();

void setup() {
   lcd.begin(16, 2);
   pinMode(ledPin, OUTPUT);
   pinMode(buttonPin, INPUT);
   Serial.begin(9600);
   printLcd();
}

void loop(){
   // verifica se o usuario quer calibrar
       os sensores
   buttonDetect();

   // verifica os sensores
   readSensors();
   delay(50);

   // entrada de 1 pessoa
   if(order.equals("1-2")){
      people++;
      order = "";
      lastChar = "";
      hasNumPersonChanged = true;
   }
   // saida de 1 pessoa
   else if(order.equals("2-1") && people >
       0){
      people--;
      order = "";
      lastChar = "";
      hasNumPersonChanged = true;
   }

   if (hasNumPersonChanged) { // Mudou-se
       o contador
      if (people > maxPeople) { //
          ultrapassou a capacidade maxima
         hasReachedMax = true;
```

```
            if (hadPreviousReachedMax ==
                false){ // Primeira vez
                ultrapassando a capacidade
                maxima
               lcd.clear();
               lcd.setCursor(3,0); //
                   centraliza o texto
               lcd.print("Atencao!!!");
               lcd.setCursor(0,1);
               lcd.print("Limite Atingido!");
                   // texto centralizado
               firstReachedMax = true;
               firstReachedMax_LCDPrintTime =
                   millis() + 1000;
            }
            if (firstReachedMax == false)
                reachedMaxPrint();
            hadPreviousReachedMax = true;
         } else { // nao ultrapassou a
             capacidade maxima
            ledState = 0;
            digitalWrite(ledPin, ledState);
                // desliga o led do 7 para
                parar de piscar
            hasReachedMax = false;
            printLcd();
            hadPreviousReachedMax = false;
         }
         hasNumPersonChanged = false;
      }
      if(hasReachedMax) blinkLed(); // O Led
          conectado ao pino 7 sempre pisca
          depois que ultrapasosu a capacidade
          maxima
      if(firstReachedMax){ // ReachedMaxPrint
          com 1s de delay na primeira vez que
          ocorrer a ultrapassagem
         if(millis() >
             firstReachedMax_LCDPrintTime){
            reachedMaxPrint();
            firstReachedMax = false;
         }
      }
   }


   void buttonDetect(){
      // Como nao requer que o botao fique
          desligado entre os 2 clicks, segurar
          o botao tambem funciona
      buttonState = digitalRead(buttonPin);
      if (buttonState == 1){ // o botao foi
          pressionado
         delay(100);

         if(numClicks == 0){ // primeiro click
            timePressed = millis();
            timePressedLimit = timePressed +
                1000;
            numClicks = 1;
         }

         else if (numClicks == 1 && millis()
             < timePressedLimit){ // segundo
             click dentro do tempo
            calibrate(); // ativa a calibragem

            // resetar as variaveis
            timePressed = 0;
            timePressedLimit = 0;
            numClicks = 0;
         }

         else if (numClicks == 1 && millis()
             > timePressedLimit){ // segundo
             click fora do tempo
            timePressed = 0;
            timePressedLimit = 0;
            numClicks = 0;
         }
      }
   }

   void calibrate(){
      // imprime a mensagem indicando o
          inicio da calibragem
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Iniciando");
      lcd.setCursor(0, 1);
      lcd.print("Calibragem");
      printTimer();

      // Calibragem da Luminosidade do
          Ambiente
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Luminosidade");
      lcd.setCursor(0, 1);
      lcd.print("do Ambiente");
      buttonState = 0;
      while(buttonState == 0){ // pausa a
          execucao do codigo ate que seja
          pressionado o botao;
         buttonState = digitalRead(buttonPin);
      }
      ambientLuminosity1 =
          analogRead(ldrPin1);
      ambientLuminosity2 =
          analogRead(ldrPin2);
      //imprimir os valores lidos, pode-se
          comentar para pular essa parte
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Ambiente1: ");
```

```
lcd.print(ambientLuminosity1);
lcd.setCursor(0, 1);
lcd.print("Ambiente2: ");
lcd.print(ambientLuminosity2);
printTimer();

// Calibragem da Luminosidade do Laser
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Luminosidade");
lcd.setCursor(0, 1);
lcd.print("do Laser");
buttonState = 0;
while(buttonState == 0){ // pausa a
    execucao do codigo ate que seja
    pressionado o botao;
   buttonState = digitalRead(buttonPin);
}
laserLuminosity1 = analogRead(ldrPin1);
laserLuminosity2 = analogRead(ldrPin2);
//imprimir os valores lidos, pode-se
    comentar para pular essa parte
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Laser1: ");
lcd.print(laserLuminosity1);
lcd.setCursor(0, 1);
lcd.print("Laser2: ");
lcd.print(laserLuminosity2);
printTimer();

if (laserLuminosity1 >
    ambientLuminosity1 &&
    laserLuminosity2 >
    ambientLuminosity2){// condicao para
    garantir que a luminosidade do laser
    > ambiente
   // Determina o valor de trigger para
       cada sensor -> Media aritimetica
   triggerValue1 = (laserLuminosity1 +
       ambientLuminosity1)/2;
   triggerValue2 = (laserLuminosity2 +
       ambientLuminosity2)/2;
   //Indica o fim da calibragem
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("Trigger 1: ");
   lcd.print(triggerValue1);
   lcd.setCursor(0, 1);
   lcd.print("Trigger 2: ");
   lcd.print(triggerValue2);
   printTimer();

   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("Calibragem");
   lcd.setCursor(0, 1);
   lcd.print("Completa!");
      printTimer();

      printLcd();
      }
   else {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Calibragem");
      lcd.setCursor(0, 1);
      lcd.print("Falhou!");
      delay(1000);
      calibrate();
    }
}

void printLcd(){
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("Maximo: ");
   lcd.print(maxPeople);
   lcd.setCursor(0, 1);
   lcd.print("Pessoas: ");
   lcd.print(people);
}

void reachedMaxPrint(){
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("Perigo! Max: ");
   lcd.print(maxPeople);
   lcd.setCursor(0, 1);
   lcd.print("Pessoas: ");
   lcd.print(people);
}

void blinkLed(){
   if (millis() >= ledTime){
      ledTime = millis() + period;
      ledState = 1 - ledState;
      digitalWrite(ledPin, ledState);
   }
}

void readSensors(){
   ldrValue1 = analogRead(ldrPin1); //
       Quanto maior o valor, maior a
       incidencia de luminosidade
   ldrValue2 = analogRead(ldrPin2); //
       Quando menor o valor lido, menor a
       incidencia de luminosidade
                                                                    //
```

```
// caso o sensor LDR1 seja interceptado
    e nao esta recebendo luminosidade o
    suficiente e o sensor LDR2 esteja
    recebendo luminosidade
if (ldrValue1 <= triggerValue1 &&
    ldrValue2 > triggerValue2 &&
    lastChar != "1"){
   order += "1";
   lastChar = "1";
   Serial.println(order);
}

// caso os sensores estejam sendo
    interceptados
else if(ldrValue1 <= triggerValue1 &&
    ldrValue2 <= triggerValue2 &&
    lastChar != "-"){
   order += "-";
   lastChar = "-";
   Serial.println(order);
}

// caso o sensor LDR2 seja interceptado
    e nao esta recebendo luminosidade o
    suficiente e o sensor LDR1 esteja
    recebendo luminosidade
else if(ldrValue2 <= triggerValue2 &&
    ldrValue1 > triggerValue1 &&
    lastChar != "2"){
   order += "2";
   lastChar = "2";
   Serial.println(order);
}

// caso uma pessoa ative o primeiro
    sensor e saia
// e dps outra pessoa ative o segundo
    sensor, seja pra sair
// ou simplesmente esbarrando o order
    passa a contar dessa
// nova leitura
else if(order.length() == 2 &&
    order.charAt(1) != '-'){
   order.remove(0, 1);
}

// se order tiver 3 caracteres e nao
    indicar nem entrada
// nem saida, apaga o order e o ultimo
    digito junto
// e o numero de pessoas permanece
    inalterado
else if(order.length() >= 3){
   if(order != "1-2" || order != "2-1"){
      order = "";
      lastChar = "";
      }   1023
   }
}

void printTimer(){
   lcd.setCursor(14, 1);
   lcd.print("#");
   lcd.setCursor(15, 1);
   lcd.print(3);
   delay(1000);
   lcd.setCursor(15, 1);
   lcd.print(2);
   delay(1000);
   lcd.setCursor(15, 1);
   lcd.print(1);
   delay(1000);
}
```

B. Distribuidor automatizado de álcool

```
#include <LiquidCrystal.h>

int cm = 0;
int motor = 13;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

long readUltrasonicDistance(int
    triggerPin, int echoPin)
{
 pinMode(triggerPin, OUTPUT);
 digitalWrite(triggerPin, LOW);
 delayMicroseconds(2);
 digitalWrite(triggerPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(triggerPin, LOW);
 pinMode(echoPin, INPUT);
 return pulseIn(echoPin, HIGH);

}

void setup()
{
 Serial.begin(9600);
 pinMode(motor, OUTPUT);
 lcd.begin(16, 2);
}

void loop()
{

 cm = 0.01723 * readUltrasonicDistance(8,
     8);
 delay(1000);
 Serial.print(cm);
 Serial.println("cm");

 if(cm<=30){
  lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("Lavando...");
   digitalWrite(motor, HIGH);
   lcd.setCursor(0,1);
   lcd.print("Use mascara! :)");
   delay(5000);
   lcd.clear();
   lcd.print("Retire suas maos :)");
   digitalWrite(motor, LOW);
     delay(2000);
  }
 else{
   digitalWrite(motor, LOW);
   lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Lave suas maos");
  lcd.setCursor(0,1);
  lcd.print("Aproxime-as");
   delay(500);
 }
```

C. Contador de distanciamento em filas

```
#include <LiquidCrystal.h>

float cm,duracao, alguem;

//definindo pinos da placa LCD
    LiquidCrystal lcd(<pino RS>, <pino
    enable>,
// <pino D4>, <pino D5>, <pino D6>, <pino
    D7>)
LiquidCrystal lcd(A4, 2, 3, 4, 5, 6);

//Sensores
byte pinoTrans1 = 8; //pino trig 1
byte pinoRec1 = 7; //pino echo 1
byte pinoTrans2 = 13; //pino trig 2
byte pinoRec2 = 12; //pino echo 2

//LED
int pinoR = 10;
int pinoG = 9;
int pinoB = 11;

//Buzzer
int buzzer = A5;

void setup()
{
 //Display LCD / Colunas e Linhas
 lcd.begin(16,2);


 //LED
 pinMode(pinoR, OUTPUT);
 pinMode(pinoG, OUTPUT);
 pinMode(pinoB, OUTPUT);

 //SENSOR
 pinMode(pinoTrans1, OUTPUT);
 pinMode(pinoTrans2, OUTPUT);
 pinMode(pinoRec1, INPUT);
 pinMode(pinoRec2, INPUT);

 //BUZZER
 pinMode(buzzer, OUTPUT);


 //iniciar a porta serial
 Serial.begin(9600);

}

void loop()
{
 delay(1000);
 lcd.clear();
 delay(500);
 cm = distancia(); //distancia em cm
 alguem = distancia1();
 if (alguem > 0){ //checa se h algum a
     frente na fila
   //limpa a tela
  //acender o LED com base na distancia,
     mudando de cor.
  //distancia de 3m (distanciamento
     social! :))
  if(cm >0 && cm<=100) { //vermelho caso
     muito perto
   analogWrite(pinoR, 255);
   analogWrite(pinoG, 0);
   analogWrite(pinoB, 0);
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("AFASTE-SE");
   lcd.setCursor(0,1);
   lcd.print("POR FAVOR");
   Serial.print("AFASTE-SE POR FAVOR");
   tone(buzzer,250);
   delay(500);

  }
  else if (cm >100 && cm <=150){ //amarelo
     caso se aproximando
   analogWrite(pinoR, 255);
   analogWrite(pinoG, 255);
   analogWrite(pinoB, 0);
   lcd.clear();
   lcd.print("AFASTE-SE");
   lcd.setCursor(0,1);
   lcd.print("POR FAVOR");
   Serial.print("AFASTE-SE POR FAVOR");
   tone(buzzer,200);
   delay(500);

  }
  else if (cm>150){ //azul caso longe
   analogWrite(pinoR, 0);
   analogWrite(pinoG, 0);
   analogWrite(pinoB, 255);
   lcd.clear();
   noTone(buzzer);
   delay(500);

  }
  else{ //apagado em qualquer outro caso
   analogWrite(pinoR, 0);
   analogWrite(pinoG, 0);
   analogWrite(pinoB, 0);
   lcd.clear();
   delay(500);
   noTone(buzzer);
  }
  //printa no serial a distancia em cm
```

```
  Serial.print(cm);
  Serial.println(" cm");
  delay(500);
 }
 else{
  lcd.clear();
  analogWrite(pinoR, 0);
  analogWrite(pinoG, 0);
  analogWrite(pinoB, 0);
  lcd.setCursor(0,0);
  lcd.print("PROXIMO");
  Serial.print("PROXIMO");
  delay(500);
  noTone(buzzer);
 }
}
//funcao para calcular a distancia:
float distancia() {
 //limpar o pino/aguardar
 digitalWrite(pinoTrans2, LOW);
 delayMicroseconds(1);
 //pulso do ultrassom
 digitalWrite(pinoTrans2, HIGH);
 delayMicroseconds(10);
 digitalWrite(pinoTrans2, LOW);
 //calcula a duracao em ms para ida e volta
     do pulso
 duracao = pulseIn(pinoRec2, HIGH);
 //calculo do tempo de ida+volta / 2 *
     0.0343
 float calcDistancia = (duracao/2) * 0.0343;
 if (calcDistancia>=331){
  calcDistancia=0;
 }
 delay(500);
 return calcDistancia;
}
//funcao para checar se h algum no outro
    lugar da fila
float distancia1(){
 //limpar o pino/aguardar
 digitalWrite(pinoTrans1, LOW);
 delayMicroseconds(1);
 //pulso do ultrassom
 digitalWrite(pinoTrans1, HIGH);
 delayMicroseconds(10);
 digitalWrite(pinoTrans1, LOW);
 //calcula a duracao em ms para ida e volta
     do pulso
 duracao = pulseIn(pinoRec1, HIGH);
 //mesmo calculo anterior
 float calcDistancia = (duracao/2) * 0.0343;
 if (calcDistancia<=331){
  calcDistancia=1;
 }
 else{
  calcDistancia=0;
 }
 delay(500);
 return calcDistancia;
```