Relatório Final

Sistema Autonômo de registro de compras

Caio Costa Maciel Cardoso - 120112663
Universidade de Brasília - UNB
FGA - Campus Gama
Brasília, Brasil
caiocmcardoso@gmail.com

Victor Barreto Batalha - 130018155
Universidade de Brasília - UNB
FGA – Campus Gama
Brasília, Brasil
victor.batalha@hotmail.com

***Resumo*— Leitor de códigos de barras para carrinhos de compras utilizando uma Raspberry e uma webcam.**

Palavras-chaves— Códigos de barras, Raspberry Pi, compras.

I. INTRODUÇÃO

Algumas atividades comuns do dia-a-dia costumam demandar muito tempo, um exemplo clássico são as filas de supermercados, mesmo depois de um longo tempo escolhendo seus produtos ainda é necessário enfrentar demoradas filas para que sejam conferidos todos seus produtos.

Visando reduzir o tempo gasto nessa atividade, foi pensado em um dispositivo integrado ao carro de compras que permite analisar o preço do produto a ser comprado, ao mesmo tempo que permite acrescentar o preço do produto, aos demais do carro. Será feita a leitura do código de barras por meio de uma câmera, lendo os símbolos deste código, pela variação na largura das barras e assim interpretando o produto a ser comprado, e o valor deste.

II. OBJETIVOS

Implementar um sistema que execute a leitura de diversos códigos de barras, faça a comparação com códigos previamente cadastrados, permita que o usuário cadastre ou não o produto lido e crie uma lista com os produtos que o usuário deseja cadastrar e ainda mostre o valor total e atualizado dos itens que estão no carrinho, em um display de nokia 5110, através de um leitor de códigos de barras por câmera.

III. JUSTIFICATIVA E REQUISITOS

Observando a movimentação nos caixas em supermercados notamos a necessidade de facilitar e agilizar esse processo. Estabelecimentos que não utilizam dessa técnica já são bastante comuns na Europa por exemplo, onde os próprios clientes passam seus produtos no caixa e realizam o pagamento, ainda assim é necessário certo trabalho para registrar todos os itens que já estavam no carrinho de compras.

Buscando resolver este problema, o projeto em questão visa implementar um sistema que mostra o valor total da compra em um display no próprio carrinho, esse valor também deve ser atualizado assim que um item é adicionado, fazendo com o pagamento ao final da compra seja muito mais rápido.

A utilização de uma câmera para a identificação do código do produto é algo que deve facilitar bastante a utilização do sistema já que o processamento de imagens em uma placa como a Raspberry Pi 3 é bastante rápido.

Como requisitos do sistema, visa-se obter:

- Decodificar com eficiência os códigos de barras.
- Registrar imagens com boa nitidez para facilitar a decodificação.
- Registrar os itens e atualizar os valores no display de acordo com a necessidade do usuário de confirmação ou não.
- Fazer a soma dos valores das compras em tempo real.

IV. DESCRIÇÃO DE HARDWARE

O projeto tem como foco o processamento de imagens, dessa forma não há uma ênfase em hardware, tendo em vista que boa parte das análises são realizadas para o entendimento e o desenvolvimento em software. Dessa forma o hardware do sistema se concentra em um display Nokia 5110, que visa fazer a interface entre usuário e hardware.

A. Raspberry Pi 3 Modelo B

Sendo o componente principal, é o componente em que se encontra o processador, sendo este o sistema embarcado, que realizará o processamento de imagens dos códigos a serem traduzidos e a tomada de decisão no projeto.

B. Webcam - Sony

Componente que captura várias fotos do produto contendo os códigos de barra analisa os frames em busca do código de barras possui um sistema de foco, porém apresenta uma resolução de 8MP, suficiente para a aplicação.

C. Nokia 5110

Será utilizado para realizar a interface entre o usuário e o sistema embarcados, possibilitando visualizar os produtos comprados o valor da compra, e as telas de navegação para realizar as operações necessárias.

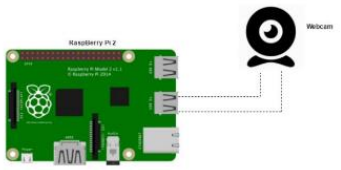

Figura 1 – Esquemático do hardware

Quantidade	Materiais
1	Raspberry Pi 3 Modelo B
1	Nokia 5110
1	Webcam Sony
1	PowerBank
2	Botões

Tabela 1 – Materiais Utilizados

V. DESCRIÇÃO DE SOFTWARE

Para implementar o projeto é utilizada a biblioteca OpenCv[3] - Open Source Computer Vision Library, essa biblioteca tem como propósito auxiliar na construção de projetos na área de visão computacional, possuindo módulos para o tratamento de fotos e vídeos, utilizando a linguagem Python. Foi implementado inicialmente o projeto em linux para verificar a funcionalidade desta em seguida os blocos de códigos da biblioteca a serem utilizados serão implementados na Raspberry Pi. Serão explicados os blocos de códigos utilizados.

O software foi dividido em blocos para que executam diferentes funções dentro do programa, permitindo dessa forma que todo o escopo do software seja executado através das funções chamadas nos momentos determinados.

Para isso foi criada primeiramente uma função chamada de "bem_vindo" que serve apenas para mostrar ao usuário que o programa está em execução. Depois foi criado uma função com o nome de "leitura", essa função, como explicita o nome, foi criada para que seja feita a iniciação da webcam e da leitura do código de barras mostrado. Dentro desse código foi criada uma função para a comparação do código de barras visualizado, sendo que o funcionamento desta, parte do principio de que o código de barras lido tenha sido previamente cadastrado, dessa forma é feita uma comparação do código lido com os códigos cadastrados, umas vez que essa série de números seja reconhecida é chamada uma nova função, chamada de "confirm_buttons", que por sua vez tem a obrigação de perguntar ao usuário se ele deseja cadastrar o produto o qual ele mostrou o código de barras ou não, a partir disso é necessário que seja escolhido o botão para confirmar (SIM) ou o botão para declinar (NÃO), chamando uma função que serve apenas para mostrar ao usuário que o produto não foi cadastrado. Caso o usuário opte por cadastrar o produto, será apresentado no display uma mensagem de que o cadastro foi efetuado e o produto adicionado à lista. Quando o usuário deseja adicionar o produto, por sua vez é chamada uma função de cadastro e escrita do nome, valor e código do produto à um arquivo de texto CSV e é chamada uma função para que seja modificado o valor total, adicionando o valor do novo produto. Feito isso, todas essas funções retornam à função leitura e espera que um novo código de barras seja mostrado, para que assim seja feito, ou não, um novo cadastro, todos os passos são repetidos enquanto o cliente desejar. Quando o usuário concluir que não deseja adicionar mais nenhum produto, ou se quer verificar o preço dos mesmos, foi criado um botão que serve para encerrar a lista. Esse botão, também é uma função, que foi nomeada de "send_email", quando esse botão é acionado durante a amostragem de que um novo código de barras é aguardado, ele faz com que o arquivo seja finalizado, e é adicionado uma última linha ao arquivo csv, linha que contém o valor total da compra efetuada, a partir daí, é mandado um email para o usuário (no caso do protótipo, uma vez que em um caso ideal de uso, essa lista seria enviada ao caixa do supermercado), desse ponto é possível fazer a verificação dos produtos escolhidos que foram adicionados a lista, fazendo assim com que o projeto funcione de acordo com o que foi proposto.

A. *Identificador de QrCode e Barcode*

Para resolver esse requisito foi utilizada a abordagem de cascade, que consiste em determinar retângulos em volta da figura analisada (código de barras ou Qrcode) e após a identificação da área onde será feito o processamento de imagem, fazer a decodificação desse código e mostrar esse código decifrado no terminal, foi dado o nome de barcode_reader.py para este código.

A leitura é feita utilizando algumas bibliotecas especializadas como a "pyzbar" e a instalação da OpenCV à raspberryPi. Após a identificação da imagem e do processamento para a conversão do código é uma string é fácil fazer a comparação do que foi lido aos códigos previamente registrados, dando assim resultados de acordo com o que se espera do software.

B. *Registro e amostragem dos produtos*

O registro é feito através de um *push_button* comum, utilizando um resistor de *pull-up* interno, ou seja, criado pelo próprio GPIO da raspberryPi. O registro é feito através da comparação do valor enviado pelo botão, como o resistor é de *pull-up* o código estará recebendo "true" como entrada enquanto o botão não for pressionado, já quando o botão é pressionado ele envia "false", nesse caso existe uma condição de comparação que quando feito e retornar uma verdade, fara o cadastro do último item lido ao arquivo csv que contem todos os dados dos produtos adicionados o carrinho e à lista. Caso o usuário deseje apenas ver o preço e não comprar o produto, basta que não aperte nenhum botão, após 5 segundos será automaticamente mostrado na tela um aviso de que o produto não foi adicionado à lista e será pedido a leitura de um novo código de barras. Dessa forma, quando o programa é encerrado, é possível que seja enviada uma lista com apenas os produtos registrados e o valor total obtido através dos cadastros de produtos realizados.

C. Imagens do Protótipo

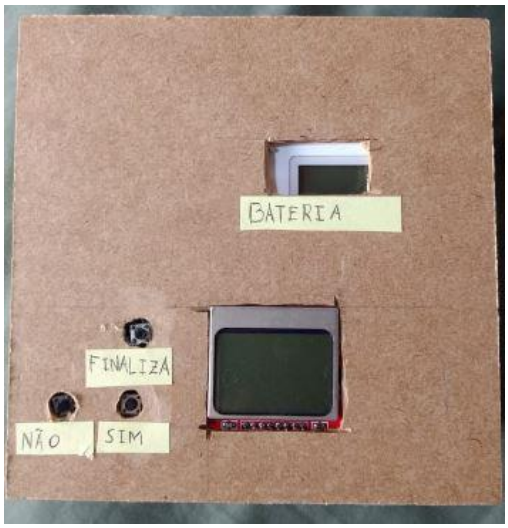

Figura 1 – Protótipo visão superior.


Figura 2 – Protótipo visão frontal.

VI. Resultados e Conclusões

Alguns problemas foram encontrados durante a execução e a programação do projeto. O primeiro obstáculo, foi lidar com a programação utilizando as bibliotecas OpenCV e pyzbar, uma vez que elas foram lidas e entendidas, os alunos obtiveram conhecimento para a criação do código base que faz a leitura e o processamento de imagem. Tendo feito isso, bastou que os alunos desenvolvessem módulos que fossem chamados durante a execução da leitura, para que fosse feita uma comunicação com o usuário. Feito isso, o último obstáculo enfrentado, foi entender e aprender como fazer a criação de um server e um login em um dos e-mails pessoais de um dos alunos para que fosse enviada a lista por email, não necessitando assim que a raspberry estivesse conectada há um monitor para que fosse mostrada a lista de compras. Feito isso, é possível ler a lista feita através do próprio célular, uma vez que a lista é enviada por email.

Para o último ponto de controle, foi pressuposto que o projeto tivesse funcionalidade através de um display e que fossem adicionados os botões para confirmação de produtos e encerramento da lista. Todos os objetivos foram cumpridos, dessa forma o projeto já possui um protótipo completo, ou seja, já funciona como o esperado, já possui uma estrutura externa e realiza todas as funções propostas pelos alunos. O projeto está, desta forma, finalizado.

Para o protótipo final foi adicionado mais um botão que permitisse o usuário a escolha de cadastrar ou não o produto, portanto não depende mais do tempo para que o leitor retorne ao seu estado de leitura, agora enquanto um dos botões não for pressionado o programa irá continuar esperando pela escolha do usuário.

Desta forma, é possível afirmar que o trabalho está finalizado e cumpre todas as funções definidas no escopo do mesmo.

Referências

[1] Contagem de objetos em movimento com OpenCV e Python usando Raspberry Pi. Dispnível em: https://www.embarcados.com.br/objetos-opencv-e-python-raspberry-pi/ . Acesso em 20 out. 2018.

[2] SIMÕES, Eduardo Dusanoski. DESENVOLVIMENTO DE SISTEMA PARA LEITURA DE CÓDIGO DE BARRAS COM ”FEEDBACK” PARA AQUISIÇÃO E SEGURANÇA DE PRODUTOS EM SUPERMERCADOS. 2015. 53 f. Disponível em: http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/6789/1/CT_CO . Acesso em: 20 out. 2018.

[3] OpenCv library. http://projectabstracts.com/list-of-projects-on-image-processing Acesso em 27 out.2018.

[4]

APÊNDICE

CÓDIGO 1 - LEITURA

```
# python barcode_scanner_video.py

# Bibliotecas
from imutils.video import VideoStream
from pyzbar import pyzbar
import argparse
import datetime
import imutils
import time
import cv2
import Adafruit_Nokia_LCD as LCD
import Adafruit_GPIO.SPI as SPI
import RPi.GPIO as GPIO
from PIL import ImageFont
from PIL import Image
from PIL import ImageDraw
import wiringpi
from email.mime.multipart import MIMEMultipart
from email.MIMEImage import MIMEImage
from email.mime.text import MIMEText
import smtplib

GPIO.setmode(GPIO.BCM)
sim = 5
nao = 6
enviar = 13
GPIO.setup(sim,GPIO.IN, pull_up_down =
GPIO.PUD_UP)
GPIO.setup(nao,GPIO.IN, pull_up_down =
GPIO.PUD_UP)
GPIO.setup(enviar,GPIO.IN, pull_up_down =
GPIO.PUD_UP)

valorf=0


nome = ('RELOGIO SAF')
valor = 349.99
nome2 = ('SKYRIM PS4')
valor2 = 169.99
nome3 = ('PS4 PRO')
valor3 = 2299.99
nome4 = ('LONG HEINEKEN')
valor4 = 3.99
nome5 = ('BBQ ZERO')
valor5 = 4.99
# Raspberry Pi hardware SPI config:
DC = 23
RST = 24
SPI_PORT = 0
SPI_DEVICE = 0

# Hardware SPI usage:
disp = LCD.PCD8544(DC, RST,
spi=SPI.SpiDev(SPI_PORT, SPI_DEVICE,
max_speed_hz=4000000))

def send_email():
    global valorf
    csv.write("\nValor Total: R$ {}\n".format(valorf))
    csv.flush()
    valorf = 0

    # Create blank image for drawing.
    # Make sure to create image with mode '1' for 1-bit color.
    image = Image.new('1', (LCD.LCDWIDTH,
LCD.LCDHEIGHT))

    # Get drawing object to draw on image.
    draw = ImageDraw.Draw(image)

    # Draw a white filled box to clear the image.

draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT)
, outline=255, fill=255)

    # Clear display.
    disp.clear()
    disp.display()
    # Load default font.
    font = ImageFont.load_default()
    # Write some text.
    draw.text((0,2), 'OBRIGADO POR', font=font)
    draw.text((0,10), 'COMPRAR', font=font)
    draw.text((0,25), 'CONOSCO!', font=font)
    # Display image.
    disp.image(image)
    disp.display()
    # Initialize library.
    disp.begin(contrast=60)
    time.sleep(4)

       # Create blank image for drawing.
    # Make sure to create image with mode '1' for 1-bit color.
    image = Image.new('1', (LCD.LCDWIDTH,
LCD.LCDHEIGHT))

    # Get drawing object to draw on image.
    draw = ImageDraw.Draw(image)
```

```
    # Draw a white filled box to clear the image.

draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT)
, outline=255, fill=255)

    # Clear display.
    disp.clear()
    disp.display()
    # Load default font.
    font = ImageFont.load_default()
    # Write some text.
    draw.text((0,2), 'LISTA', font=font)
    draw.text((0,15), 'FINALIZADA', font=font)
    # Display image.
    disp.image(image)
    disp.display()
    # Initialize library.
    disp.begin(contrast=60)
    time.sleep(3)

    # create message object instance
    msg = MIMEMultipart()

    i = 0
    # setup the parameters of the message
    password = "$$k1a2l3l4$$"
    msg['From'] = "caiocmcardoso@gmail.com"
    msg['To'] = "caiocmcardoso@gmail.com"
    msg['Subject'] = "Lista de Compras Carrinho 1"

    # attach image to message body
    msg.attach(MIMEText(file("barcodes.csv").read()))


    # create server
    server = smtplib.SMTP('smtp.gmail.com: 587')

    server.starttls()

    # Login Credentials for sending the mail
    server.login(msg['From'], password)


    # send the message via the server.
    server.sendmail(msg['From'], msg['To'], msg.as_string())

    server.quit()

    criar_novo_csv()

    # Clear display.
    disp.clear()
    disp.display()

    # Create blank image for drawing.
    # Make sure to create image with mode '1' for 1-bit color.
    image = Image.new('1', (LCD.LCDWIDTH,
LCD.LCDHEIGHT))

    # Get drawing object to draw on image.
    draw = ImageDraw.Draw(image)

    # Draw a white filled box to clear the image.

draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT)
, outline=255, fill=255)

     # Load default font.
    font = ImageFont.load_default()
    # Write some text.
    draw.text((0,2), 'LISTA', font=font)
    draw.text((0,15), 'ENVIADA', font=font)
    # Display image.
    disp.image(image)
    disp.display()
    # Initialize library.
    disp.begin(contrast=60)
    time.sleep(3)
    print "successfully sent email to %s:" % (msg['To'])

    # Create blank image for drawing.
    # Make sure to create image with mode '1' for 1-bit color.
    image = Image.new('1', (LCD.LCDWIDTH,
LCD.LCDHEIGHT))

    # Get drawing object to draw on image.
    draw = ImageDraw.Draw(image)

    # Draw a white filled box to clear the image.

draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT)
, outline=255, fill=255)

    # Clear display.
    disp.clear()
    disp.display()
    # Load default font.
    font = ImageFont.load_default()
    # Write some text.
    draw.text((1,2), 'NOVA LISTA', font=font)
    draw.text((5,15), 'CRIADA', font=font)
    # Display image.
    disp.image(image)
    disp.display()
    # Initialize library.
    disp.begin(contrast=60)
    time.sleep(4)
    bem_vindo()
    return

def criar_novo_csv ():
    filename = "barcodes.csv"
    # opening the file with w+ mode truncates the file
```

```
    f = open(filename, "w+")

def mostrar_img(bar_code):

    cod1 = '135202780730'
    cod2 = '020320001833'
    cod3 = '140419991403'
    cod4 = '020619605804'
    cod5 = '070919734505'

    # Clear display.
    disp.clear()
    disp.display()

    # Create blank image for drawing.
    # Make sure to create image with mode '1' for 1-bit color.
    image = Image.new('1', (LCD.LCDWIDTH,
LCD.LCDHEIGHT))

    # Get drawing object to draw on image.
    draw = ImageDraw.Draw(image)

    # Draw a white filled box to clear the image.

draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT)
, outline=255, fill=255)

    if bar_code == cod1:
        # Load default font.
        font = ImageFont.load_default()
        # Write some text.
        draw.text((1,2), nome, font=font)
        draw.text((5,15), 'R$ 349,99', font=font)
        # Display image.
        disp.image(image)
        disp.display()
        # Initialize library.
        disp.begin(contrast=60)
        time.sleep(4)
        confirm_buttons(bar_code, nome, valor)

    if bar_code == cod2:
        # Load default font.
        font = ImageFont.load_default()
        # Write some text.
        draw.text((1,2), nome2, font=font)
        draw.text((5,15), 'R$ 169,99', font=font)
        # Display image.
        disp.image(image)
        disp.display()
        # Initialize library.
        disp.begin(contrast=60)
        time.sleep(4)
        confirm_buttons(bar_code, nome2, valor2)

    if bar_code == cod3:
        # Load default font.
        font = ImageFont.load_default()
        # Write some text.
        draw.text((1,2), nome3, font=font)
        draw.text((5,15), 'R$ 2299,99', font=font)
        # Display image.
        disp.image(image)
        disp.display()
        # Initialize library.
        disp.begin(contrast=60)
        time.sleep(4)
        confirm_buttons(bar_code, nome3, valor3)

    if bar_code == cod4:
        # Load default font.
        font = ImageFont.load_default()
        # Write some text.
        draw.text((1,2), nome4, font=font)
        draw.text((5,15), 'R$ 3,99', font=font)
        # Display image.
        disp.image(image)
        disp.display()
        # Initialize library.
        disp.begin(contrast=60)
        time.sleep(4)
        confirm_buttons(bar_code, nome4, valor4)

    if bar_code == cod5:
        # Load default font.
        font = ImageFont.load_default()
        # Write some text.
        draw.text((1,2), nome5, font=font)
        draw.text((5,15), 'R$ 4,99', font=font)
        # Display image.
        disp.image(image)
        disp.display()
        # Initialize library.
        disp.begin(contrast=60)
        time.sleep(4)
        confirm_buttons(bar_code, nome5, valor5)

    bem_vindo()
    return

def confirm_buttons (barcodeData, nomep, valorp):

    # Clear display.
    disp.clear()
    disp.display()

    # Create blank image for drawing.
    # Make sure to create image with mode '1' for 1-bit color.
    image = Image.new('1', (LCD.LCDWIDTH,
LCD.LCDHEIGHT))

    # Get drawing object to draw on image.
    draw = ImageDraw.Draw(image)
```

```
    # Draw a white filled box to clear the image.

draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT)
, outline=255, fill=255)

    # Load default font.
    font = ImageFont.load_default()
    # Write some text.
    draw.text((0,0), 'VOCE DESEJA', font=font)
    draw.text((0,10), 'ADICIONAR', font=font)
    draw.text((0,20), 'O PRODUTO', font=font)
    draw.text((0,30), 'A SUA LISTA?', font=font)
    # Display image.
    disp.image(image)
    disp.display()
    # Initialize library.
    disp.begin(contrast=60)

    cont=0
    while cont==0:
        input_state1=GPIO.input(5)
        input_state2=GPIO.input(6)
        if input_state1 == False:
            confirma_prod(valorp)
            csv.write("{}, {}, {}\n".format(nomep, valorp,
barcodeData))
            csv.flush()
            found.add(barcodeData)
            cont=1

        if input_state2 == False:
            nao_confirma_prod()
            cont=1

    return

def bem_vindo():
    # Clear display.
    disp.clear()
    disp.display()

    # Create blank image for drawing.
    # Make sure to create image with mode '1' for 1-bit color.
    image = Image.new('1', (LCD.LCDWIDTH,
LCD.LCDHEIGHT))

    # Get drawing object to draw on image.
    draw = ImageDraw.Draw(image)

    # Draw a white filled box to clear the image.

draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT)
, outline=255, fill=255)

    # Load default font.
    font = ImageFont.load_default()
    # Write some text.
    draw.text((0,0), 'BEM VINDO', font=font)
    draw.text((0,10), 'AGUARDANDO', font=font)
    draw.text((0,20), 'CODIGO', font=font)
    draw.text((0,30), 'DE BARRAS', font=font)
    # Display image.
    disp.image(image)
    disp.display()
    # Initialize library.
    disp.begin(contrast=60)

def confirma_prod(valorp):
    # Clear display.
    disp.clear()
    disp.display()

    # Create blank image for drawing.
    # Make sure to create image with mode '1' for 1-bit color.
    image = Image.new('1', (LCD.LCDWIDTH,
LCD.LCDHEIGHT))

    # Get drawing object to draw on image.
    draw = ImageDraw.Draw(image)

    # Draw a white filled box to clear the image.

draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT)
, outline=255, fill=255)

    # Load default font.
    font = ImageFont.load_default()
    # Write some text.
    draw.text((0,0), 'PRODUTO', font=font)
    draw.text((0,10), 'ADICIONADO', font=font)
    draw.text((0,20), 'A LISTA', font=font)
    # Display image.
    disp.image(image)
    disp.display()
    # Initialize library.
    disp.begin(contrast=60)
    time.sleep(3)
    imagem_total(valorp)
    return

def nao_confirma_prod():
    # Clear display.
    disp.clear()
    disp.display()

    # Create blank image for drawing.
    # Make sure to create image with mode '1' for 1-bit color.
    image = Image.new('1', (LCD.LCDWIDTH,
LCD.LCDHEIGHT))

    # Get drawing object to draw on image.
    draw = ImageDraw.Draw(image)

    # Draw a white filled box to clear the image.
```

```
draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT)
, outline=255, fill=255)

    # Load default font.
    font = ImageFont.load_default()
    # Write some text.
    draw.text((0,0), 'PRODUTO NAO', font=font)
    draw.text((0,10), 'ADICIONADO', font=font)
    draw.text((0,20), 'A LISTA', font=font)
    # Display image.
    disp.image(image)
    disp.display()
    # Initialize library.
    disp.begin(contrast=60)
    time.sleep(4)
    return

def imagem_total(valorp):

    global valorf
    valorf = valorf + valorp
    value = str(valorf)

    # Clear display.
    disp.clear()
    disp.display()

    # Create blank image for drawing.
    # Make sure to create image with mode '1' for 1-bit color.
    image = Image.new('1', (LCD.LCDWIDTH,
LCD.LCDHEIGHT))

    # Get drawing object to draw on image.
    draw = ImageDraw.Draw(image)

    # Draw a white filled box to clear the image.

draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT)
, outline=255, fill=255)

    # Load default font.
    font = ImageFont.load_default()
    # Write some text.
    draw.text((0,0), 'VALOR TOTAL', font=font)
    draw.text((0,10), value, font=font)
    # Display image.
    disp.image(image)
    disp.display()
    # Initialize library.
    disp.begin(contrast=60)
    time.sleep(3)
    return

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-o", "--output", type=str,
default="barcodes.csv",
            help="path to output CSV file containing
barcodes")
args = vars(ap.parse_args())

# Inicializando a camera
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
#Para cameraPI descomentar linha abaixo e comentar acima
#vs = VideoStream(usePiCamera=True).start()
time.sleep(2.0)

# Criando um arquivo CSV para armazsenar os dados
# Codigos salvos
csv = open(args["output"], "w")
found = set()


def leitura():

    bem_vindo()
    # loop over the frames from the video stream
    while True:
        # Redimensionando as dimensoes de imagem para
largura maxima de
        # 400 pixels
        frame = vs.read()
        frame = imutils.resize(frame, width=400)

        # achando o codigo de barras no frame e
decodificando cada um deles
        barcodes = pyzbar.decode(frame)

        # loop pelos codigos detectados
        for barcode in barcodes:
            # extraindo os limites da caixa de localizacao
dos codigos para
            # desenhar as caixas no codigo de barras da
imagem
            (x, y, w, h) = barcode.rect
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0,
255), 2)


            # o codigo de barras e um objeto de data entao
para desenhar
            # na nossa saida de video precisamos
transformar em uma string
            barcodeData = barcode.data.decode("utf-8")
            barcodeType = barcode.type

            # desenhando o codigo de barras e a caixa na
imagem
            text = "{} ({})".format(barcodeData,
barcodeType)
            cv2.putText(frame, text, (x, y - 10),
```

```
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(0, 0, 255), 2)
                mostrar_img(barcodeData)


        # mostrado o frame na saida
        cv2.imshow("Barcode Scanner", frame)
        key = cv2.waitKey(1) & 0xFF
        envia=GPIO.input(enviar)
        if envia == 0:
           send_email()

        # se a teclar q for pressionada acaba o loop
        if key == ord("q"):
            break

    # fechando o arquivo CSV
    print("[INFO] cleaning up...")
    csv.close()
    cv2.destroyAllWindows()
    vs.stop()

leitura()
```

CÓDIGO 2 – Cadastro de Produtos

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
#define MAX 150
#define MINCODIGO 100


/*Objetivo: Dar as opcoes ao usuario de cadastrar produtos no arquivo, editar os ja cadastrados, ver os produtos
  que ja foram cadastrados, apagar todos os arquivos ou sair do programa.
  Entrada: Um numero que selecione a opcao desejada pelo usuario. Caso, seja cadastrar mais produtos, os dados do produto:
  nome, preço, código. Caso seja edicao, um numero que selecione qual dado do
  produto quer editar.
  Saida: Se o usuario selecionar a opcao de visualizar os arquivos que ja foram cadastrados, mostre em uma lista, todos os dados
  de todos os produtos cadastrados: nome, preco e código. */

typedef struct produto
{
    char nome[MAX];
    float valor;
    long int codigos_de_barras;
    char opcao;
} dados ;

int main()
{
    //Declaracoes
    dados cadastro;
    dados outro;
    FILE *pCadastro;
    dados *auxiliar;
    char opcao, opcao2;
    int escolha, aux, i=0, procuracodigo=0;
    long int codigoauxiliar;

    //Prototipos
    void corrigeString(char *pString);
    void validaOpcao(char *pOpcao);
    void validaCodigo(long int *pCodigo);
    void ordemAlfabetica(dados *pStruct, int i);
    void validaEscolha(int *pEscolha);


    do
    {
        //Menu
        printf("BEM VINDO AO CADASTRO DE PRODUTOS!\n==========================================\n");
        printf("O que voce deseja fazer?\n1 - Cadastrar novo produto\n2 - Editar dados do produto\n3 - Consultar todos os produtos\n4 - Apagar arquivo de dados\n5 - Sair do programa\n");
        scanf("%d", &escolha);
        fflush(stdin);
        system("CLS");

        switch (escolha)
        {
            case 1:
                if((pCadastro = fopen("Produtos.bin", "a+b")) == NULL)
                {
                    printf("Arquivo nao pode ser aberto!\n");
                    exit(1);
                }
                else
                {
                    //Cadastrando a matricula
                    printf("Insira o codigo de barras: \n");
                    scanf("%ld", &cadastro.codigos_de_barras);
                    validaMatricula(&cadastro.codigo_de_barras);
                    system("CLS");
                    fflush(stdin);

                    //Cadastrando o nome
                    printf("Insira o nome do produto: \n");
                    fgets(cadastro.nome, MAX, stdin);
                    corrigeString(cadastro.nome);
                    system("CLS");


                    //Cadastrando o valor
                    printf("Insira o valor do produto: \n");
                    scanf("%f", &cadastro.preco);
                    validaAltura(&cadastro.preco);
                    fflush(stdin);
                    system("CLS");

                    if((fwrite(&cadastro, sizeof(cadastro), 1, pCadastro)) != 1)
                    {
                        printf("Arquivo nao armazenado! \n");
                        exit(1);
                    }
                    fclose(pCadastro);
                } break;

            case 2:
                do
```

```
        {
            i=0;
            opcao2= 'S';
            procuramatricula = 0;
            printf("Insira o codigo de barras que deseja editar: \n");
            scanf("%ld", &codigoauxiliar);
            if((pCadastro = fopen("Produtos.bin", "r+b")) == NULL)
            {
               printf("\aOcorreu um erro, ou o arquivo nao existe!\n");
               exit(1);
            }
            else
            {
               do
               {
                  fread(&outro, sizeof(dados), 1, pCadastro);
                  if(codigoauxiliar == outro.ccodigo)
                  {
                     fflush(stdin);
                     printf("Esse e o produto que deseja editar?S(sim)/N(nao) \n\n");
                     printf("%30s %10s %10s %7s %10s\n", "NOME", "CODIGO DE BARRAS", "VALOR");
                     printf("%30s %10s %10ld %7.2f %10s\n", outro.nome, outro.codigos_de_barras, outro.valor, (outro.opcao == 'N') ? "PRIMEIRA" : "OUTRA");
                     opcao2 = getchar();
                     opcao2 = toupper(opcao2);
                     validaOpcao(&opcao2);
                     procuracodigo = 1;
                     system("CLS");
                     if(opcao2 == 'S')
                     {
                        do
                        {
                           printf("O que voce deseja editar?\n1 - Nome\n2 - Codigo de Barras\n3 - Valor\n4 - Tudo\n");
                           scanf("%d", &escolha);
                           fflush(stdin);
                           system("CLS");
                           switch(escolha)
                           {
                           case 1:
                              printf("Insira o novo nome do Produto: \n");
                              fgets(cadastro.nome, MAX, stdin);
                              corrigeString(cadastro.nome);
                              fseek(pCadastro, i*sizeof(dados), SEEK_SET);
                              fwrite(&cadastro.nome, sizeof(cadastro.nome), 1, pCadastro);
                              system("CLS");
                              break;

                           case 2:
                              printf("Insira o novo Codigo de barras do Produto: \n");
                              fgets(cadastro.codigos_de_barras, MAX, stdin);
                              corrigeString(cadastro.codigos_de_barras);
                              fseek(pCadastro, (i*sizeof(dados)+sizeof(cadastro.nome)), SEEK_SET);
                              fwrite(&cadastro.pais, sizeof(cadastro.codigos_de_barras), 1, pCadastro);
                              system("CLS");
                              break;

                           case 3:
                              printf("Insira o novo valor do Produto: \n");
                              scanf("%f", &cadastro.valor);
                              validaAltura(&cadastro.valor);
                              fseek(pCadastro, (i*sizeof(dados)+sizeof(cadastro.nome)+sizeof(cadastro.valor)), SEEK_SET);
                              fwrite(&cadastro.valor, sizeof(cadastro.valor), 1, pCadastro);
                              system("CLS");
                              break;

                           case 4:
                              printf("Insira o novo nome do Produto: \n");
                              fgets(cadastro.nome, MAX, stdin);
                              corrigeString(cadastro.nome);
                              system("CLS");

                              printf("Insira o novo Codigo de Barras do Produto: \n");
                              fgets(cadastro.codigos_de_barras, MAX, stdin);
                              corrigeString(cadastro.codigos_de_barras);
                              fflush(stdin);
                              system("CLS");

                              printf("Insira o novo valor do produto: \n");
                              scanf("%f", &cadastro.valor);
                              validaAltura(&cadastro.valor);
                              fflush(stdin);
                              system("CLS");

                              fseek(pCadastro, i*sizeof(dados), SEEK_SET);
```

```
                        fwrite(&cadastro, sizeof(dados), 1, pCadastro);
                        system("CLS");
                      }
                    printf("Voce deseja editar mais algum dado?S(sim)/N(nao)\n");
                    fflush(stdin);
                    opcao2 = getchar();
                    opcao2 = toupper(opcao2);
                    validaOpcao(&opcao2);
                    system("CLS");
                    if(opcao2 == 'S')
                       rewind(pCadastro);
                    } while(escolha < 1 || escolha > 4 || opcao2 == 'S');
                  }
                }
                i++;
              } while(!feof(pCadastro) && opcao2 == 'S');
              fclose(pCadastro);
              if(procuracodigo == 0)
                  printf("Codigo de Barras nao encontrada!\n");
              printf("Voce deseja editar os dados de outra Codigo?S(sim)/N(nao)\n");
              fflush(stdin);
              opcao2 = getchar();
              opcao2=toupper(opcao2);
              validaOpcao(&opcao2);
              system("CLS");
            }
          } while(opcao2 == 'S');
          break;

        case 3:
          i=0;
          if((pCadastro = fopen("Produtos.bin", "r+b")) == NULL)
          {
            printf("\aOcorreu um erro, ou o arquivo nao existe!\n");
            exit(1);
          }
          else
          {
            while(!feof(pCadastro))
              {
                fread(&outro, sizeof(dados), 1, pCadastro);
                if(!feof(pCadastro))
                  i++;
                printf("%d", i);
              }
              if(i==0)
                printf("Nenhum nome cadastrado!\n");
              else
              {
                if((auxiliar = (dados *)calloc(i,sizeof(dados))) == NULL)
                {
                    printf("Ocorreu um erro!");
                    exit(1);
                }
                else
                {
                  aux=0;
                  rewind(pCadastro);
                  while(!feof(pCadastro))
                  {
                    fread(&auxiliar[aux], sizeof(dados), 1, pCadastro);
                    aux++;
                  }
                  printf("Voce deseja visualizar o arquivo em ordem alfabetica\n1 - Crescente(A-Z)\n2 - Descrescente(Z-A)\n");
                  scanf("%d", &escolha);
                  validaEscolha(&escolha);
                  ordemAlfabetica(auxiliar, i);
                  system("CLS");
                  printf("%30s %10s %10s %7s %10s\n", "NOME", "VALOR", "CODIGO DE BARRAS";
                  switch (escolha)
                  {
                  case 1:
                    for(aux=0;aux<i;aux++)
                    {
                      printf("%30s %10s %10ld %7.2f %10s\n", auxiliar[aux].nome, auxiliar[aux].valor, auxiliar[aux].codigos_de_barras);
                    }
                  break;
                  case 2:
                    for(aux=i-1;aux>=0;aux--)
                    {
                      printf("%30s %10s %10ld %7.2f %10s\n", auxiliar[aux].nome, auxiliar[aux].valor, auxiliar[aux].codigos_de_barras);
                    }
                  }
                  free(auxiliar);
                  fclose(pCadastro);
                }
              }
          }
          getch();
          system("CLS");
          break;

        case 4:
          printf("Voce tem certeza que deseja APAGAR TODO O ARQUIVO? S(sim)/N(nao)\n");
          opcao = getchar();
          opcao = toupper(opcao);
```

```
            validaOpcao(&opcao);
            if(opcao == 'S')
            {
               if((pCadastro = fopen("Produtos.bin", "wb"))
== NULL)
                  {
                     printf("Arquivo nao pode ser apagado!");
                     fclose(pCadastro);
                  }
               else
               {
                  printf("Arquivo apagado com sucesso!\n");
                  fclose(pCadastro);
               }
            }
            else
            {
               printf("Arquivo nao foi apagado!\n");
               fclose(pCadastro);
            }
            getch();
            system("CLS");
            break;
      }
   }while(escolha != 5);
   printf("Obrigado e volte sempre!\n");
}


//Subprogramas
/*Objetivo: Corrigir e validar as Strings
  Parametro: Uma string
  Retorno: nenhum.*/
void corrigeString(char *pString)
{
   if(pString[strlen(pString)-1] == '\n')
      pString[strlen(pString)-1] = '\0';
   while(pString[0] == '\0' || pString[0] == ' ')
   {
      printf("Dado invalido! Insira novamente: \n");
      fgets(pString, MAX, stdin);
      if(pString[strlen(pString)-1] == '\n')
         pString[strlen(pString)-1] = '\0';
   }
}

/*Objetivo: Validar as opcoes
  Parametro: Uma string
  Retorno: nenhum.*/
void validaOpcao (char *pOpcao)
{
   fflush(stdin);
   while (*pOpcao != 'S' && *pOpcao != 'N')
   {
      printf("Opcao invalida! Digite S(sim)/N(nao)\n");
      *pOpcao = getchar();
      *pOpcao = toupper(*pOpcao);
      fflush(stdin);
   }
}

/*Objetivo: Validar a matricula e garantir que nao seja
repitida
  Parametro: A matricula
  Retorno: nenhum.*/
void validaMatricula (long int *pCodigo)
{
   dados outro;
   int repete;
   FILE *pArquivo;

   while(*pCodigo < MINCODIGO)
   {
      printf("Matricula invalida! Insira novamente: \n");
      scanf("%ld", pCodigo);
   }

   if((pArquivo = fopen("Jogadores.bin", "rb")) == NULL)
   {
      printf("Ocorreu um erro no cadastro!\n");
      exit(1);
   }
   else
   {
      do
      {
         repete = 0;
         while(!feof(pArquivo))
         {
            fread(&outro, sizeof(outro), 1, pArquivo);
            if(*pCodigo == outro.codigos_de_barras)
            {
               printf("Codigo de barras ja cadastrado! Insira
novamente: \n");
               scanf("%ld", pCodigo);
               rewind(pArquivo);
               repete = 1;
            }
         }
      }while(repete == 1);
      fclose(pArquivo);
   }
}

/*Objetivo: Validar a altura
  Parametro: A altura
  Retorno: nenhum.*/
void validaAltura (float *pValor)
{
   while(*pAltura < 0)
   {
      printf("Valor invalido! Insira valor novamente: \n");
      scanf("%f", pValor);
   }
```

```
}

/*Objetivo: Validar a escolha
  Parametro: A escolha
  Retorno: nenhum.*/
void validaEscolha(int *pEscolha)
{
   while(*pEscolha < 1 || *pEscolha > 2)
   {
      printf("Opcao invalida! Digite:\n1 - Crescente\n2 -
Descrescente\n");
      scanf("%d", pEscolha);
   }
}

/*Objetivo: Colocar em ordem alfabetica
  Parametro: Uma matriz de struct com o tamanho dos dados
ja cadastrados e o contador com o tamanho da struct
  Retorno: nenhum.*/
void ordemAlfabetica(dados *pStruct, int i)
{
        int aux, aux2;
        dados auxiliar;

            for( aux = 0; aux <= i - 1; aux++)
            {
                for( aux2 = aux + 1; aux2 <= i; aux2++)
                {

if(strcmp(pStruct[aux2].nome,pStruct[aux].nome) < 0)
                        {
                                auxiliar        =
pStruct[aux2];
                                pStruct[aux2]   =
pStruct[aux];
                                pStruct[aux] = auxiliar;
                        }

                }

            }
}
```