AI & SUSTAINABILITY IN VET EDUCATION

Nome do(s) autor(es): Sucessos Criativos, Lda

Título do WP: WP3 – IA-SAVES@ProjectHub

Referência de Atividade: 3.2 – Modelo de projeto para IA-SAVES@ProjectHub

Logótipo:

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

Índice

1. Título do Projeto: 4
2. Categoria do Projeto: 4
3. Descrição do Projeto: 4
4. Palavras-chave: 4
5. Público-alvo: 4
6. Objetivos do Projeto: 4
7. Cronologia do Projeto: 5
8. Recursos Fornecidos: 5
9. Ferramentas e Tecnologias Utilizadas: 5
10. Plano de Implementação Passo a Passo: 6
11. Critérios de Avaliação: 23
12. Línguas Disponíveis: 23
13. Colaboradores: 23
14. Feedback e Melhoria Contínua: 24

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

1. Título do Projeto:

Sistema de Self-Checkout Baseado em IA com Reconhecimento de Produtos

---

2. Categoria do Projeto:

IA em Robótica

---

3. Descrição do Projeto:

Este projeto implementa um protótipo de sistema de self-checkout baseado em IA que utiliza visão computacional para reconhecer automaticamente os produtos colocados em frente a uma câmara e calcular o seu preço total. O sistema foi concebido para simular uma experiência simplificada de comércio self-service, onde os utilizadores controlam manualmente o processo de digitalização usando botões físicos.

Quando um produto é colocado à vista da câmara, um modelo de deteção de objetos identifica itens suportados, como frutas, legumes e alimentos embalados. Ao pressionar um botão "Adicionar", todos os itens válidos atualmente detetados no fotograma da câmara são adicionados a um inventário virtual. Este processo pode ser repetido várias vezes, permitindo aos utilizadores digitalizar produtos em lotes. Depois de adicionados todos os artigos, um botão "Checkout" mostra o resumo final da compra e o preço total. Um botão "Reiniciar" limpa o inventário, preparando o sistema para o próximo cliente. Este projeto demonstra como a visão alimentada por IA, os controlos de hardware e as interfaces de utilizador podem ser combinados para criar sistemas inteligentes de automação do retalho.

---

4. Palavras-chave:

Raspberry Pi, Deteção de Objetos, Automação do Retalho, Sistemas Inteligentes

---

5. Público-alvo:

Tanto Educadores como Alunos

---

6. Objetivos do Projeto:

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

- Detetar e reconhecer produtos de retalho usando uma câmara e um modelo de deteção de objetos.
- Permitir que os utilizadores controlem manualmente a digitalização de itens usando botões físicos.
- Mantenha um inventário virtual com contagem de itens e preços.
- Calcule e mostre o preço total de compra em tempo real.
- Fornecer um fluxo de trabalho claro de checkout e reset para vários utilizadores.
- Explore conceitos de automação impulsionada por IA aplicados a ambientes de retalho.

---

7. Cronologia do Projeto:

Com os recursos fornecidos, este projeto deverá demorar aproximadamente 6-7 horas ou menos a ser concluído de forma calma.

---

8. Recursos Fornecidos:

- Código Python a implementar deteção de objetos, gestão de inventário e lógica de checkout
- Vídeo demo do resultado
- Diagrama de cablagem para botões, ecrã LCD e câmara

---

9. Ferramentas e Tecnologias Utilizadas:

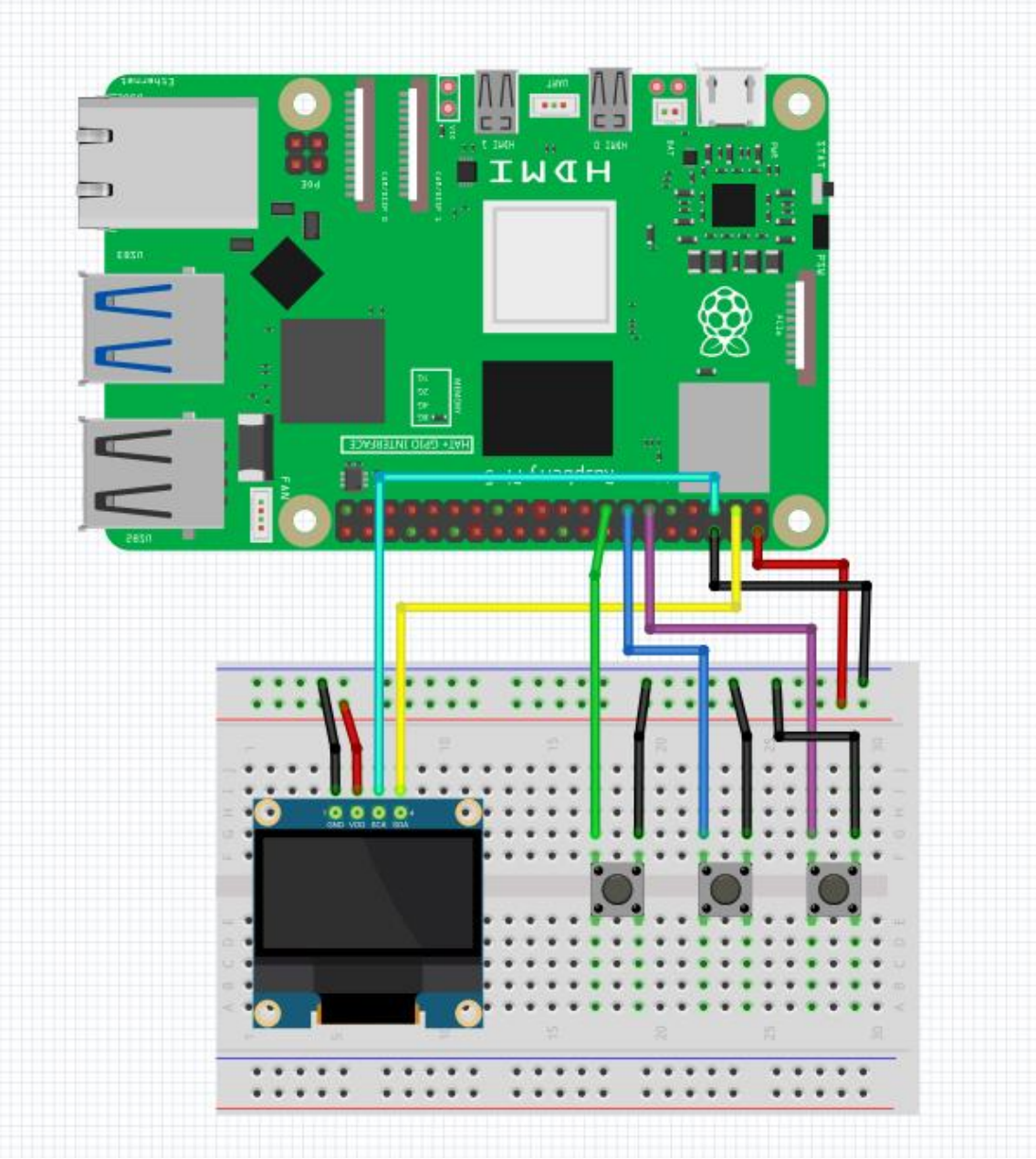
- Raspberry Pi 5
- Módulo da Câmara Raspberry Pi 4 v2
- Linguagem de programação Python
- OpenCV para entrada de câmara e processamento de imagem
- Modelo de Deteção de Objetos Pré-treinado (YOLO11-Nano)
- Ecrã OLED (SSD1306) para feedback do utilizador
- Botões físicos para interação com o utilizador
- Bibliotecas GPIO para controlo de entrada por hardware

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

10. Plano de Implementação Passo a Passo:

1. **Preparação de Hardware:** Ligue o Raspberry Pi, o módulo da câmara, os botões e o ecrã LCD de acordo com o diagrama de cablagem.



2. **Configuração de Software:** Instale o software necessário no Raspberry Pi, incluindo bibliotecas Raspbian OS e Python, incluindo OpenCV, Picamera2, adafruit_ssd1306, Ultralytics YOLO e GPIO.

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

3. **Execute o código:** Copie o código fornecido abaixo e guarde-o no seu Raspberry Pi.

```
Importado CV2

Tempo de importação

Importação de threading

de picamera2 importado Picamera2

da importação de ultralytics YOLO

importar RPi.GPIO como GPIO

Tabuleiro de importação

Importação adafruit_ssd1306

a partir de PIL importar Image, ImageDraw, ImageFont


# ===== CONFIGURAÇÃO DE PREÇOS (EDITAR AQUI) =====

PREÇOS = {

    "maçã": 0,50,

    "banana": 0,30,

    "laranja": 0,40,

    "Sandes": 2,50,

    "Brócolos": 1,20,

    "cenoura": 0,60

}


# Lista de objetos detetáveis (só estes mostram as caixas)

DETECTABLE_OBJECTS = ["maçã", "banana", "laranja", "sandes", "brócolos",
"cenoura"]


# ===== CONFIGURAÇÃO DO PINO =====

BUTTON_ADD = 17 # Botão para adicionar itens
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
BUTTON_FINISH = 27 # Botão para terminar/finalizar

BUTTON_RESET = 22 # Botão para reiniciar inventário


# ===== CONFIGURAÇÃO GPIO =====

GPIO.setmode(GPIO. BCM)

GPIO.setup(BUTTON_ADD, GPIO. DENTRO, pull_up_down=GPIO. PUD_UP)

GPIO.setup(BUTTON_FINISH, GPIO. DENTRO, pull_up_down=GPIO. PUD_UP)

GPIO.setup(BUTTON_RESET, GPIO. DENTRO, pull_up_down=GPIO. PUD_UP)


# Rastreamento de ressalto

last_button_time = {"add": 0, "finish": 0, "reset": 0}

DEBOUNCE_TIME = 0,5 # segundos


# ===== CONFIGURAÇÃO DE ECRÃ OLED =====

LARGURA = 128

ALTURA = 64


# Inicializar o ecrã I2C e OLED

i2c = tabuleiro. I2C()

OLED = adafruit_ssd1306. SSD1306_I2C(LARGURA, ALTURA, i2c, addr=0x3C)


# Ecrã limpo

oled.fill(0)

oled.show()


# Carregar fonte

Experimente:
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
    font                                                                    =
ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 10)

    font_small                                                              =
ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 8)

    font_large                                                              =
ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans-Bold.ttf",
12)

Exceto:

    fonte = ImageFont.load_default()

    font_small = ImageFont.load_default()

    font_large = ImageFont.load_default()


# ===== CÂMARA + CONFIGURAÇÃO YOLO =====

picam2 = Picamera2()

picam2.preview_configuration.main.size = (640, 400)

picam2.preview_configuration.main.format = "RGB888"

picam2.preview_configuration.align()

picam2.configure("pré-visualização")

picam2.start()


modelo = YOLO("yolo11n.pt")


# ===== SISTEMA DE INVENTÁRIO =====

inventário = {} # {item_name: contagem}

system_state = "PRONTO" # Indica: PRONTO, A DIGITALIZAR, FINALIZAR

scroll_offset = 0

scroll_direction = 1

camera_running = Verdadeiro

latest_frame = Nenhum

latest_results = Nenhum
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
# ===== FUNÇÕES DE EXIBIÇÃO =====

Definitivamente getfontsize(fonte, texto):

    """Calcular o tamanho do texto em pixels"""

    esquerda, cima, direita, baixo = font.getbbox(texto)

    voltar direita - esquerda, baixo - cima



def update_display():

    """Atualizar o ecrã OLED com base no estado atual"""

    scroll_offset global, scroll_direction


    # Cria uma imagem em branco para desenhar

    imagem = imagem.new("1", (oled.width, oled.height))

    draw = ImageDraw.Draw(image)


    se system_state == "PRONTO":

        # Título

        título = "VENDEDOR INTELIGENTE"

        (tw, th) = getfontsize(font_large, título)

        draw.text((oled.width // 2 - tw // 2, 2), título, fonte=font_large,
preenchimento=255)


        # Instruções

        draw.text((2,    20),    "Pressione    ADICIONAR    para    escanear",
font=font_small, preenchimento=255)

        draw.text((2, 32), "items on table", font=font_small, fill=255)


        # Separador de linha
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
        draw.line((0, 44, 128, 44), preencher=255)



        # Mensagem pronta

        draw.text((10,  50),  "Pronto  para  começar!",  fonte=font_small,
preenchimento=255)



    elif system_state == "A DIGITALIZAR":

        # Título

        draw.text((5,    0),    "SCANNING    MODE",    fonte=font_small,
preenchimento=255)

        draw.line((0, 12, 128, 12), fill=255)



        # Lista de inventário com scroll

        Se não for o inventário:

            draw.text((5,  18),  "Ainda  sem  itens",  fonte=font_small,
preenchimento=255)

            draw.text((5,  30),  "Colocar  item  e",  fonte=font_small,
preenchimento=255)

            draw.text((5,    40),    "preme    ADDIR",    fonte=font_small,
preenchimento=255)

        Caso mais:

            # Calcular a altura total do conteúdo

            total_items = len (inventário)

            item_height = 10

            total_height = total_items * item_height



            # Desenhar itens com deslocamento de rolo

            y_start = 15 - scroll_offset

            Para IDX, (item, contagem) em enumerate(inventory.items()):

                y_pos = y_start + (idx * item_height)
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
                # Só desenha se for visível no ecrã
                se -10 <= y_pos <= 64:
                    price = PRICES.get(item, 0.0)
                    item_total = preço * contagem
                    texto      =      f"{item.capitalize()}:      {count}x
€{item_total:.2f}"
                    draw.text((2, y_pos), text, font=font_small, fill=255)

            # Tira o total no fundo (sempre visível)
            draw.rectangle((0, 54, 128, 64), fill=0, outline=255)
            total = sum(PRICES.get(item, 0) * contagem para item, contagem
em inventário.items())
            draw.text((2,  55),  f"Total:  €{total:.2f}",  font=font_small,
fill=255)

            # Atualizar scroll para efeito de loop contínuo
            se total_height > 40: # Só deslocar se o conteúdo for maior do
que a área de exibição
                scroll_offset += scroll_direction
                Se scroll_offset > total_height:
                    scroll_offset = 0
                Elif scroll_offset < 0:
                    scroll_offset = total_height

    elif system_state == "CHECKOUT":
        # Título
        título = "CHECKOUT"
        (tw, th) = getfontsize(font_large, título)
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
        draw.text((oled.width // 2 - tw // 2, 2), título, fonte=font_large,
preenchimento=255)

        draw.line((0, 14, 128, 14), preencher=255)



        Se não for o inventário:

            draw.text((25, 30), "Sem itens!", fonte=fonte, filho=255)

        Caso mais:

            # Resumo do programa

            total_items = som(inventário.valores())

            total_price = sum(PRICES.get(item, 0) * conta por item, conta em
inventário.items())



            draw.text((5,  18),  f"Items:  {total_items}",  fonte=font_small,
preenchimento=255)

            draw.line((0, 30, 128, 30), fill=255)

            draw.text((5,      35),      f"TOTAL:      €{total_price:.2f}",
font=font_large, fill=255)

            draw.line((0, 48, 128, 48), preencher=255)

            draw.text((15,    52),    "Prema    RESETAR",    fonte=font_small,
preenchimento=255)



    # Mostrar a imagem em OLED

    oled.imagem(imagem)

    oled.show()




def display_thread():

    """Thread de atualização contínua do ecrã"""

    embora Verdadeiro:

        Experimente:
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
            update_display()
            time.sleep(0.1) # Atualização 10 vezes por segundo para scroll
suave
        exceto exceção como e:
            print(f"Erro de visualização: {e}")
            tempo.dormir(0.5)



def camera_preview_thread():
    """Pré-visualização contínua da câmara com deteção YOLO"""
    latest_frame global, latest_results, camera_running

    Enquanto camera_running:
        Experimente:
            # Fotograma de captura
            frame = picam2.capture_array()

            # Executar deteção YOLO
            resultados = model(frame, verbose=False)

            # Armazenar resultados para uso pela função add_items
            latest_frame = frame.copy()
            latest_results = resultados

            # Criar moldura anotada para exposição
            annotated_frame = frame.copy()

            # Caixas de deteção de puxada
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
            para resultados em resultados:

                Para a caixa em Result.Boxes:

                    cls_id = int(box.cls[0])

                    label = result.names[cls_id].lower()

                    confiança = box.conf[0] * 100


                    # Apenas caixas de exposição para objetos detetáveis

                    se o rótulo em DETECTABLE_OBJECTS:

                        x1, y1, x2, y2 = map(int, box.xyxy[0])


                        # Escolhe a cor com base no limiar de confiança

                        Se a confiança > 50:

                            # Verde para itens que serão adicionados

                            box_color = (0, 255, 0)

                            text_color = (0, 255, 0)

                            estado = ""

                        Caso mais:

                            # Amarelo/Laranja para baixa confiança (não será
adicionado)

                            box_color = (0, 165, 255) # Laranja

                            text_color = (0, 165, 255)

                            estado = " (Demasiado Baixo)"


                        # Desenhar a caixa delimitadora

                        cv2.retângulo(annotated_frame, (x1, y1), (x2, y2),
box_color, 2)


                        # Criar texto de etiqueta (nome do artigo, confiança,
preço e estado)
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
                        price = PRICES.get(label, 0.0)
                        label_text = f"{label.capitalize()} {confiança:.1f}%
- {preço:.2f}eur{status}"

                        # Etiqueta de desenho
                        cv2.putText(
                            annotated_frame,
                            label_text,
                            (x1, y1 - 10),
                            CV2. FONT_HERSHEY_SIMPLEX,
                            0.5,
                            text_color,
                            2
                        )

            # Adicionar texto informativo no topo (sem fundo)
            # Estado
            status_text = f"Estado: {system_state}"
            cv2.putText(
                annotated_frame,
                status_text,
                (10, 25),
                CV2. FONT_HERSHEY_SIMPLEX,
                0.7,
                (0, 255, 255),
                2
            )
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
            # Texto da dica
            cv2.putText(
                annotated_frame,
                "Gorjeta: >50% de confiança necessária para adicionar ao carrinho",
                (10, 50),
                CV2. FONT_HERSHEY_SIMPLEX,
                0.45,
                (200, 200, 200), # Cinzento claro
                1
            )

            # Adicionar contagem de inventário
            Se Inventário:
                total_items = som(inventário.valores())
                total_price = sum(PRICES.get(item, 0) * conta por item, conta em inventário.items())
                inventory_text = f"Carrinho: {total_items} itens | Total: {total_price:.2f}eur"
                cv2.putText(
                    annotated_frame,
                    inventory_text,
                    (10, 80),
                    CV2. FONT_HERSHEY_SIMPLEX,
                    0.6,
                    (255, 255, 0),
                    2
                )
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
            # Quadro de exibição
            cv2.imshow ("Smart Seller - Pré-visualização em direto",
annotated_frame)

            # Pequeno atraso
            if cv2.waitKey(1) & 0xFF == ord('q'):
                camera_running = Falso
                Pausa

        exceto exceção como e:
            print(f"Erro de câmara: {e}")
            tempo.dormir(0.1)



# ===== MANUSEADORES DE BOTÕES =====
Def button_pressed(button_name):
    """Verifica se o botão foi pressionado com o debouncing"""
    current_time = tempo.tempo()
    Se current_time - last_button_time[button_name] < DEBOUNCE_TIME:
        return False
    last_button_time[button_name] = current_time
    regresso Verdade



def add_items():
    """Escanear e adicionar itens ao inventário"""
    system_state global, scroll_offset, latest_frame latest_results
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
    print("\n> A ESCANEAR PARA ITENS...")
    system_state = "DIGITALIZAÇÃO"
    scroll_offset = 0

    # Usar o fotograma mais recente e os resultados da pré-visualização da
câmara
    se latest_frame for Nenhum ou latest_results for Nenhum:
        print ("X Câmara não pronta, por favor aguarde...")
        Regresso

    items_found = []

    # Processar todas as deteções a partir de resultados em cache
    Para o resultado em latest_results:
        Para a caixa em Result.Boxes:
            cls_id = int(box.cls[0])
            label = result.names[cls_id].lower()

            # Verifica se o item está na nossa lista de objetos detetáveis
            se o rótulo em DETECTABLE_OBJECTS:
                confiança = box.conf[0] * 100

                # Só conta se a confiança for razoável
                Se a confiança > 50:
                    items_found.append(label)

    # Atualizar inventário
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
    Se items_found:
        Para o item em items_found:
            inventário[item] = inventário.obter(item, 0) + 1

        print(f"✓ Adicionado: {', '.join(items_found)}")
        print(f"Atual inventário: {inventário}")
    Caso mais:
        print("✗ Nenhum item válido detetado")




def finish_checkout():
    """Finalizar e mostrar o checkout final"""
    system_state global

    imprimir("\n> CHECKOUT")
    system_state = "CHECKOUT"

    Se Inventário:
        total_items = som(inventário.valores())
        total_price = sum(PRICES.get(item, 0) * conta por item, conta em
inventário.items())

        print("="*40)
        imprimir("RECIBO FINAL")
        print("="*40)
        Para item, contar no inventário.items():
            price = PRICES.get(item, 0.0)
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
            item_total = preço * contagem

            print(f"{item.capitalize():15}    {count}x    €{price:.2f}    =
€{item_total:.2f}")

        print("-"*40)

        print(f"{'TOTAL':15} €{total_price:.2f}")

        print("="*40)

    Caso mais:

        print ("Sem itens no carrinho!")



def reset_system():

    """Reiniciar inventário e regressar ao estado de prontidão"""

    Inventário global, system_state scroll_offset


    imprimir("\n> RESET DO SISTEMA")

    inventary.clear()

    system_state = "PRONTO"

    scroll_offset = 0

    print ("Inventário limpo. Pronto para novo cliente.")



# ===== PROGRAMA PRINCIPAL =====

print("="*50)

Impressão (" SISTEMA DE VENDEDORES INTELIGENTES)

print("="*50)

print("\nLista de Preços:")

para item, preço em PRICES.items():

    print(f" {item.capitalize():15} €{price:.2f}")
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
print("\n" + "="*50)

print("CONTROLS:")

print(f" GPIO {BUTTON_ADD:2d} - Adicionar itens ao carrinho")

print(f" GPIO {BUTTON_FINISH:2d} - Finalizar e finalizar")

print(f" GPIO {BUTTON_RESET:2d} - Sistema de reset)

print("="*50)

print("\nPress Ctrl+C para sair\n")


# Iniciar tópico de atualização do ecrã

display_updater = enfiamento. Thread(target=display_thread, daemon=True)

display_updater.start()


# Iniciar tópico de pré-visualização da câmara

camera_previewer    =    enfiamento.    Thread(target=camera_preview_thread,
daemon=True)

camera_previewer.start()


print("> Pré-visualização da câmara iniciada. Carrega em 'q' na janela de
pré-visualização para sair.")


Experimente:

    Enquanto camera_running:

        # Confirmar o botão ADICIONAR

        if GPIO.input(BUTTON_ADD) == GPIO. BAIXO:

            se  button_pressed("adicionar")  e  system_state  em  ["READY",
"SCANNING"]:

                add_items()


        # Botão de verificar TERMINAR
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
        if GPIO.input(BUTTON_FINISH) == GPIO. BAIXO:
            if button_pressed ("terminar") e system_state == "SCANNING":
                finish_checkout()

        # Verificar botão RESET
        if GPIO.input(BUTTON_RESET) == GPIO. BAIXO:
            se button_pressed("reset"):
                reset_system()

        tempo.sono(0.01)

exceto o KeyboardInterrupt:
    print("\n> Programa interrompido pelo utilizador")

Finalmente:
    print("> A limpar...")
    camera_running = Falso
    time.sleep(0.5) # Dar tempo aos fios para pararem
    GPIO.cleanup()
    cv2.destroyAllWindows()
    picam2.stop()
    oled.fill(0)
    oled.show()
    print ("Programa > terminado")
```

11. Critérios de Avaliação:

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

- **Funcionalidade do Sistema:** Capacidade de replicar o projeto.
- **Criatividade:** Sugestões de melhorias (novas funcionalidades, novos objetos detetados, etc.). Se for sugerido e tentado (com sucesso ou fracasso), deve ser mais recompensado.
- **Precisão da deteção:** Identificação correta dos produtos suportados com confiança suficiente.
- **Estabilidade do Sistema:** Funcionamento suave da pré-visualização da câmara, atualizações de ecrã e lógica de threading.

---

12. Línguas Disponíveis:

Inglês

---

13. Colaboradores:

- **Fátima Pais, Luís Soares** – Designers e Facilitadores de Projetos

---

14. Feedback e Melhoria Contínua:

Descreva como o feedback será recolhido dos participantes e como esse feedback será usado para melhorar futuros projetos ou iterações deste projeto.

- **Inquérito Pós-Implementação:** Após a implementação do projeto, o feedback será recolhido através de inquéritos direcionados aos participantes envolvidos no processo de testes e observação.
- **Sessões de Reflexão da Equipa:** A equipa do projeto organizará sessões internas de revisão para discutir os desafios enfrentados, o que funcionou bem e as áreas que precisam de aperfeiçoamento.
- **Avaliação Técnica:** Observações relativas à precisão da deteção, estabilidade do sistema e durabilidade do hardware serão documentadas para melhorar versões futuras.
- **Atualizações Iterativas:** Com base no feedback e desempenho, serão implementadas atualizações ao código do software e ao design do sistema para melhorar a fiabilidade e a eficácia.

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.