```
"""
================================================================================
=====
                        NOME DO FICHEIRO:
ArmController_PT.py
================================================================================
=====

DESCRIÇÃO PARA PRINCIPIANTES:
Este programa corre num Raspberry Pi 5 (um pequeno computador).
Ele usa uma câmara para ver o movimento do braço e da mão de uma pessoa e
traduz esses movimentos para comandos que o robô consegue entender.

O que este programa faz:
1. Captura vídeo da câmara (Raspberry Pi Camera Module 3).
2. Usa "Inteligência Artificial" (MediaPipe) para detetar onde estão os dedos, o
cotovelo e o ombro.
3. Faz cálculos matemáticos para perceber se o braço está esticado ou dobrado, e
para onde está a apontar.
4. Envia essas informações por comunicação em série (RX/TX) para o Arduino, que
controla os motores.

LINGUAGEM: Python
================================================================================
=====
"""

# --- IMPORTAR BIBLIOTECAS (Ferramentas já feitas) ---
import cv2                  # "OpenCV": Ferramenta para processar imagens e vídeo
import mediapipe as mp      # "MediaPipe": A ferramenta da Google que deteta mãos
e corpos
import math                 # Ferramentas de matemática (ângulos, distâncias)
import statistics           # Para calcular médias
import time                 # Para controlar o tempo (pausas, medir atrasos)
import serial               # Para falar com o Arduino por cabo USB/Série
from picamera2 import Picamera2 # Biblioteca específica para controlar a câmara
do Raspberry Pi 5 (Pi Camera Module 3)
import numpy as np          # Ferramenta para cálculos matemáticos avançados

# ----------------- CONFIGURAÇÃO OTIMIZADA PARA O RASPBERRY PI ----------------
-
# Definimos o tamanho da imagem. Imagens mais pequenas são processadas mais
depressa.
FRAME_WIDTH = 640   # Largura em pixels
FRAME_HEIGHT = 480  # Altura em pixels
```

```
FPS_TARGET = 30     # Objetivamos 30 imagens por segundo (fluidez normal de
vídeo)

# --- CONFIGURAÇÃO DA COMUNICAÇÃO (SERIE) ---
# Aqui dizemos onde o Arduino está ligado e a que velocidade falar com ele.
SERIAL_PORT = "/dev/ttyAMA0"  # Porta interna do Raspberry Pi (Pinos GPIO)
BAUD_RATE = 115200            # Velocidade da conversa (bits por segundo). Tem de
ser igual no Arduino!
SEND_INTERVAL = 0.1           # Só enviamos dados a cada 0.1 segundos (10
vezes/segundo) para não entupir

# ---------- PARÂMETROS AJUSTÁVEIS (Afinações) ----------
# Limiares: São valores de referência. Se passarmos deste valor, considerada-se
"Aberto" ou "Fechado".

# Dedos: Ângulo mínimo para considerar o dedo ABERTO.
TH_POLEGAR = 150.0
TH_INDICADOR = 160.0
TH_MEDIO = 150.0
TH_ANELAR = 150.0
TH_MINDINHO = 140.0

# Braço: Ângulos para saber para onde o braço aponta.
TH_BRACO_ESQUERDA = 70    # 0 a 70 graus: Braço virado para o corpo (Esquerda)
TH_BRACO_CENTRO_MAX = 130 # 70 a 130 graus: Braço em frente
                          # 130 a 180 graus: Braço para fora (Direita)

# Flexão: Diferença de altura (Y) entre o pulso e o ombro para saber se o braço
está levantado.
TH_FLEXAO_Y = 0.1  # Se a diferença for maior que 0.1, consideramos "Fletido"
# -------------------------------------------------------------

# --- INICIALIZAR O MEDIAPIPE (A Inteligência Artificial) ---
mp_pose = mp.solutions.pose       # Módulo para detetar corpo (ombros, cotovelos)
mp_hands = mp.solutions.hands     # Módulo para detetar mãos/dedos
mp_drawing = mp.solutions.drawing_utils # Útil para desenhar os "esqueletos" no
ecrã

# Configuração visual das linhas que vamos desenhar no ecrã
drawing_spec_thin = mp_drawing.DrawingSpec(thickness=1, circle_radius=1)
drawing_spec_conn = mp_drawing.DrawingSpec(thickness=1)

# Definição dos Dedos: Dicionário que diz quais os pontos (landmarks) formam cada
dedo
FINGERS = {
```

```
    "polegar": [1, 2, 4],     # Pontos específicos da mão que formam o polegar
    "indicador": [5, 6, 8],
    "medio": [9, 10, 12],
    "anelar": [13, 14, 16],
    "mindinho": [17, 18, 20]
}

# Associação dos limiares a cada dedo
FINGER_THRESHOLDS = {
    "polegar": TH_POLEGAR,
    "indicador": TH_INDICADOR,
    "medio": TH_MEDIO,
    "anelar": TH_ANELAR,
    "mindinho": TH_MINDINHO
}

# -------- INICIAR LIGAÇÃO AO ARDUINO --------
try:
    arduino = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=1)
    time.sleep(2) # Esperar 2 segundos para o Arduino acordar bem
    print(f"Ligado ao Arduino em {SERIAL_PORT}")
except Exception as e:
    # Se falhar (cabo desligado?), avisa mas não para o programa
    print(f"Não foi possível ligar ao Arduino: {e}")
    print(f"   Verifica a porta. Podes usar: ls /dev/tty*")
    arduino = None

# -------- FUNÇÕES DE LÓGICA (O Cérebro do Programa) --------

def angulo_3pts(a, b, c):
    """
    Calcula o ângulo entre 3 pontos (como um cotovelo: ombro, cotovelo, pulso).
    Recebe: Pontos a, b, c (coordenadas x,y).
    Devolve: Ângulo em graus.
    """
    # Matemática vetorial (cálculo de diferenças e produtos internos)
    # Não precisas de perceber a matemática complexa aqui, basta saber que
devolve um ângulo.
    ax, ay = a[0]-b[0], a[1]-b[1]
    cx, cy = c[0]-b[0], c[1]-b[1]
    num = ax*cx + ay*cy
    den = math.hypot(ax, ay) * math.hypot(cx, cy) + 1e-8
    cosv = max(-1.0, min(1.0, num/den))
    ang = math.degrees(math.acos(cosv))
    return ang
```

```python
def finger_angle_open(hand_landmarks, ids):
    """
    Vê se um dedo está esticado ou dobrado.
    Mede o ângulo na 'dobradiça' do meio do dedo.
    """
    a = hand_landmarks.landmark[ids[0]]
    b = hand_landmarks.landmark[ids[1]]
    c = hand_landmarks.landmark[ids[2]]
    return angulo_3pts((a.x,a.y),(b.x,b.y),(c.x,c.y))

def hand_center(hand_landmarks):
    """Calcula o ponto central da mão (média de todos os pontos)."""
    xs = [lm.x for lm in hand_landmarks.landmark]
    ys = [lm.y for lm in hand_landmarks.landmark]
    return (statistics.mean(xs), statistics.mean(ys))

def pair_hand_to_pose_side(hand_cx, hand_cy, pose_landmarks):
    """
    Descobre se a mão que estamos a ver é a DIREITA ou a ESQUERDA.
    Como faz? Vê qual dos pulsos do corpo (detetado pelo Pose) está mais perto da
mão.
    """
    if pose_landmarks is None:
        return None
    lw = pose_landmarks.landmark[15] # Pulso esquerdo do corpo
    rw = pose_landmarks.landmark[16] # Pulso direito do corpo

    # Calcular distância (Teorema de Pitágoras)
    dl = math.hypot(hand_cx - lw.x, hand_cy - lw.y)
    dr = math.hypot(hand_cx - rw.x, hand_cy - rw.y)

    return "Left" if dl < dr else "Right" # Retorna "Left" (Esq) ou "Right" (Dir)

def determinar_orientacion_brazo(angulo):
    """Diz para onde o braço está a apontar com base no ângulo do ombro."""
    if angulo <= TH_BRACO_ESQUERDA:
        return "Esquerda (para o corpo)"
    elif angulo <= TH_BRACO_CENTRO_MAX:
        return "Centro (frente)"
    else:
        return "Direita (para fora)"

def determinar_flexion_brazo(hombro_y, muneca_y):
    """
```

```
    Diz se o braço está levantado (fletido).
    Compara a altura (Y) do ombro e do pulso.
    """
    diff_y = abs(hombro_y - muneca_y)
    if diff_y > TH_FLEXAO_Y:
        return "Fletido", diff_y
    else:
        return "Estendido", diff_y

def calculate_hand_rotation(hand_landmarks, flex_status, arm_orientation):
    """
    A função mais complexa: Tenta adivinhar se a palma da mão está virada para a
câmara ou de costas.
    Usa a posição do Polegar (Thumb) e do Mindinho (Pinky).

    Devolve um valor entre -1.0 (Palma para a câmara) e 1.0 (Costas da mão /
Dorso).
    """
    thumb = hand_landmarks.landmark[4]  # Ponto na ponta do polegar
    pinky = hand_landmarks.landmark[20] # Ponto na ponta do mindinho

    diff = 0
    t_val = 0
    p_val = 0
    axis_used = "Indef"

    # Dependendo se o braço está fletido ou esticado, a lógica muda (porque a mão
roda anatomicamente).
    is_flexed = (flex_status == "Fletido")

    if is_flexed:
        # CASO 1: Braço Fletido (Dobrado)
        # Usamos a diferença no eixo X (Horizontal)
        t_val = thumb.x
        p_val = pinky.x
        diff = -(t_val - p_val)
        axis_used = "X (Fletido)"

    else:
        # Braço Estendido
        if "Direita" in arm_orientation:
            # CASO 3: Esticado para a Direita -> Usamos eixo Y (Vertical)
            t_val = thumb.y
            p_val = pinky.y
            diff = (t_val - p_val)
```

```
            axis_used = "Y (Ext-Dir)"

        else:
            # CASO 2: Esticado para o Centro/Corpo -> Usamos eixo Y Invertido
            t_val = thumb.y
            p_val = pinky.y
            diff = -(t_val - p_val)
            axis_used = "Y-Inv (Ext-Corpo)"

    # Normalização: Converte a diferença bruta em pixels num valor padrão entre -1 e 1
    MAX_DIFF = 0.22
    rotation_value = max(-1.0, min(1.0, diff / MAX_DIFF))

    # Classificar por texto para ajudar no Debug
    orientation = "Indefinido"
    if rotation_value < -0.3:
        orientation = "Palma (Camara)"
    elif rotation_value > 0.3:
        orientation = "Dorso (Camara)"
    else:
        orientation = "A rodar / Canto"

    return diff, rotation_value, orientation, axis_used

# -------- INICIALIZAÇÃO DO LOOP (CICLO) DA CÂMARA E MODELOS --------

# 1. Configurar e ligar a câmara
print("A inicializar Picamera2...")
picam2 = Picamera2()
config = picam2.create_preview_configuration(
    main={"size": (FRAME_WIDTH, FRAME_HEIGHT), "format": "RGB888"},
    controls={"FrameRate": FPS_TARGET}
)
picam2.configure(config)
picam2.start()

print("Câmara iniciada. A aguardar estabilização...")
time.sleep(2)

# 2. Configurar os modelos de deteção (MediaPipe)
pose = mp_pose.Pose(
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5,
    model_complexity=1
```

```
)
hands = mp_hands.Hands(
    max_num_hands=2,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5,
    model_complexity=1
)

# Variáveis de controlo
frame_count = 0             # Conta quantos frames já passaram
start_time = time.time()    # Guarda a hora de início
last_send = time.time()     # Guarda a hora do último envio para o Arduino

print("Sistema iniciado. Pressiona Ctrl+C para sair.")

try:
    # --- CICLO INFINITO (Repete para cada imagem da câmara) ---
    while True:
        # 1. Capturar imagem
        frame = picam2.capture_array()

        # 2. Processar com Inteligência Artificial
        pose_res = pose.process(frame)   # Procura corpo
        hands_res = hands.process(frame) # Procura mãos

        # Cria uma cópia da imagem para desenhar por cima (out)
        out = frame
        h, w = frame.shape[:2]

        # Preparar variáveis vazias (reset)
        ang_brazo = None
        orientacion_brazo = "Indefinido"
        estado_flexion = "Estendido"
        side = None
        hombro_y = 0
        muneca_y = 0
        diff_y = 0

        hand_rotation_val = 0
        hand_orientation_str = ""

        # --- PARTE A: ANÁLISE DO CORPO (POSE) ---
        if pose_res.pose_landmarks:
            plm = pose_res.pose_landmarks.landmark
```

```
            # Tenta descobrir qual é o braço ativo usando a mão detetada
            if hands_res.multi_hand_landmarks:
                hand_lm_check = hands_res.multi_hand_landmarks[0]
                hc_x, hc_y = hand_center(hand_lm_check)
                side = pair_hand_to_pose_side(hc_x, hc_y,
pose_res.pose_landmarks)

            # Se não tiver a certeza, assume direita
            if side is None:
                side = "Right"

            # Seleciona os pontos certos (Ombro, Cotovelo, Pulso) dependendo do
lado
            if side == "Left": # Pontos 11, 13, 15 são do lado esquerdo
                sh = plm[11]; el = plm[13]; wr = plm[15]
            else:              # Pontos 12, 14, 16 são do lado direito
                sh = plm[12]; el = plm[14]; wr = plm[16]

            # Calcula ângulos e lógica
            ang_brazo = angulo_3pts((sh.x, sh.y), (el.x, el.y), (wr.x, wr.y))
            orientacion_brazo = determinar_orientacion_brazo(ang_brazo)
            hombro_y = sh.y; muneca_y = wr.y
            estado_flexion, diff_y = determinar_flexion_brazo(hombro_y, muneca_y)

            # Escreve textos no ecrã para vermos o que ele está a pensar
            cv2.putText(out, f"Braco ({side}): {int(ang_brazo)} deg",
                        (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 1)
            cv2.putText(out, f"Orient: {orientacion_brazo}",
                        (10, 50), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 255),
1)
            cv2.putText(out, f"Flex: {estado_flexion}",
                        (10, 70), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 150, 0),
1)

            # Desenha o esqueleto no ecrã
            mp_drawing.draw_landmarks(out, pose_res.pose_landmarks,
mp_pose.POSE_CONNECTIONS,
                                      drawing_spec_thin, drawing_spec_conn)

        # --- PARTE B: ANÁLISE DA MÃO ---
        estados_dedos = {}
        found_hand = False

        if hands_res.multi_hand_landmarks:
            found_hand = True
```

```
            hand_lm = hands_res.multi_hand_landmarks[0] # Usa a primeira mão
encontrada

            # Calcular Rotação da mão (Palma/Dorso)
            _, hand_rotation_val, hand_orientation_str, _ =
calculate_hand_rotation(
                hand_lm, estado_flexion, orientacion_brazo)

            # Calcular se cada dedo está aberto ou fechado
            for name, ids in FINGERS.items():
                ang = finger_angle_open(hand_lm, ids)
                umbral = FINGER_THRESHOLDS[name]
                abierto = ang > umbral # Verdadeiro se Ângulo > Limite
                estados_dedos[name] = (abierto, ang)

            # --- DESENHOS DE DEBUG (Ajuda visual) ---
            # Barra de rotação
            bar_x, bar_y, bar_w, bar_h = 300, 30, 150, 15
            cv2.rectangle(out, (bar_x, bar_y), (bar_x + bar_w, bar_y + bar_h),
(100, 100, 100), 2)
            norm_pos = int((hand_rotation_val + 1) / 2 * bar_w) # Posicionar o
ponto na barra
            cv2.circle(out, (bar_x + norm_pos, bar_y + int(bar_h/2)), 6, (0, 255,
255), -1)
            cv2.putText(out, f"Rot: {hand_rotation_val:.2f}", (bar_x, bar_y-5),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 1)

            # Lista dos dedos no canto do ecrã
            y0 = 100
            ordered_keys = ["indicador", "medio", "anelar", "mindinho",
"polegar"]
            for i, k in enumerate(ordered_keys):
                abierto, ang = estados_dedos[k]
                color = (0, 255, 0) if abierto else (0, 0, 255) # Verde se
aberto, Vermelho se fechado
                cv2.putText(out, f"{k}: {int(ang)}", (10, y0 + i*20),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 1)

            # Desenha os elos da mão
            mp_drawing.draw_landmarks(out, hand_lm, mp_hands.HAND_CONNECTIONS,
                                      drawing_spec_thin, drawing_spec_conn)

        # --- PARTE C: ENVIAR COMANDOS PARA O ARDUINO ---
        # Só envia se passou tempo suficiente (SEND_INTERVAL) e se temos o
Arduino ligado
```

```
        if time.time() - last_send > SEND_INTERVAL and arduino and found_hand:

            # 1. Converter Base/Ombro para números (0, 1, 2)
            val_orient = 1 # Centro (Padrão)
            if "Esquerda" in orientacion_brazo: val_orient = 0
            elif "Direita" in orientacion_brazo: val_orient = 2

            # 2. Converter Flexão para número (0 ou 1)
            val_flex = 1 if estado_flexion == "Fletido" else 0

            # 3. Converter lista de dedos para 0s e 1s
            d_list = []
            ordered_keys = ["indicador", "medio", "anelar", "mindinho",
"polegar"]
            for k in ordered_keys:
                if k in estados_dedos:
                    d_list.append(1 if estados_dedos[k][0] else 0)
                else:
                    d_list.append(0)

            # 4. Converter rotação (-1.0 a 1.0) para graus (0 a 180) para o servo
            rot_deg = int((hand_rotation_val + 1.0) * 90)
            rot_deg = max(0, min(180, rot_deg)) # Forçar limites entre 0 e 180

            # 5. Criar a mensagem de texto
            # Formato: "$orient,flex,d1,d2,d3,d4,d5,rot\n"
            dedos_str = ",".join(map(str, d_list))
            msg = f"${val_orient},{val_flex},{dedos_str},{rot_deg}\n"

            try:
                arduino.write(msg.encode()) # Enviar pelo cabo
                last_send = time.time()
                # print(f"TX: {msg.strip()}") # (Opcional: Ver o que enviámos)
            except Exception as e:
                print(f"Erro na Série: {e}")

        # --- PARTE D: MOSTRAR IMAGEM ---
        cv2.imshow("RPi Hand Tracking", out)

        # Se carregar na tecla ESC (código 27), sair do programa
        if cv2.waitKey(1) & 0xFF == 27:
            break

        # --- ESTATÍSTICAS (FPS) ---
        frame_count += 1
```

```
        if frame_count % 30 == 0:
            elapsed = time.time() - start_time
            # Imprime na consola a velocidade (FPS) e rotação atual
            print(f"FPS: {frame_count/elapsed:.1f} | Rot:
{hand_rotation_val:.2f}")

except KeyboardInterrupt:
    print("\nA parar (Ctrl+C pressionado)...")

finally:
    # Limpeza final: Desligar câmara, fechar janelas e ligação Arduino
    picam2.stop()
    if arduino:
        arduino.close()
    cv2.destroyAllWindows()
    print("Fim.")
```