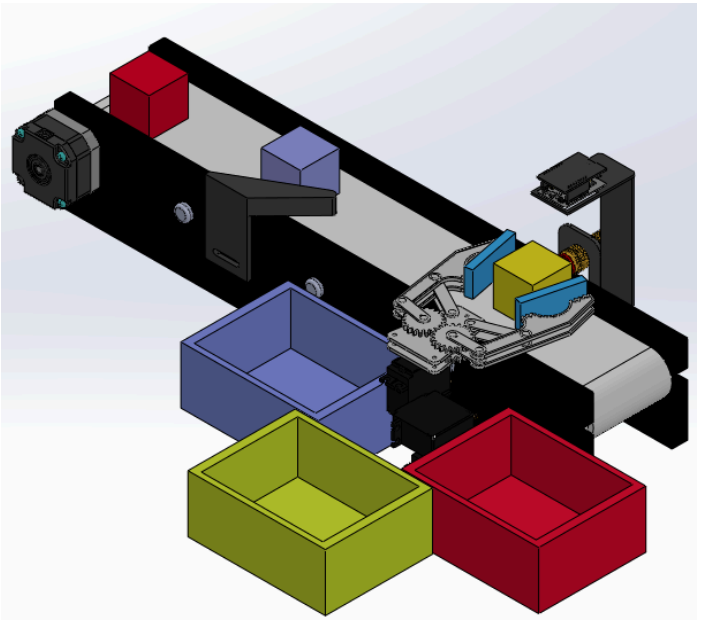
Ministério da Educação
Instituto Federal do Espírito Santo
Campus Serra

Roteiro Prático - Kit de Visão Computacional

Disciplina: Processamento Digital de Imagens

Autor: Igor Rodrigues Cassimiro



- Este kit didático foi criado com intuito de aprendizado de visão computacional
- Reproduz uma linha de produção

Objetivo:

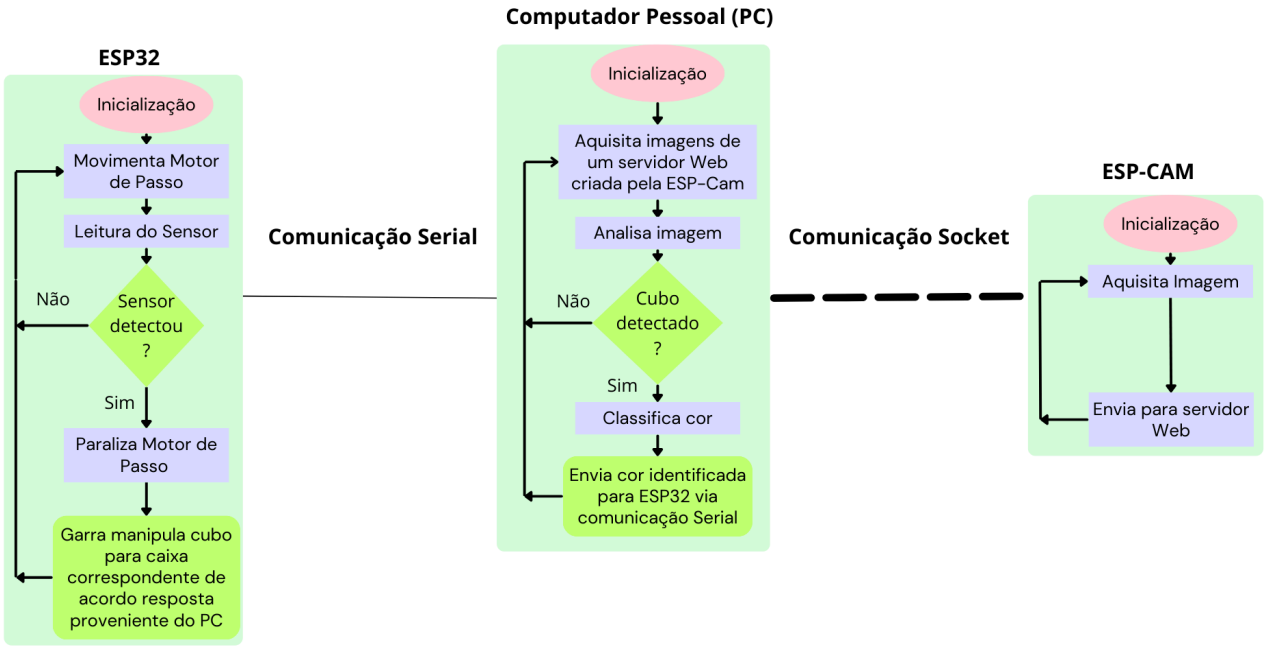
- Criar um código que reconheça a cor do cubo utilizando conceitos estudados na disciplina de Processamento Digital de Imagens.

Materiais necessários para prática:

- Notebook (PC)
- Kit de Visão Computacional

Softwares necessários:

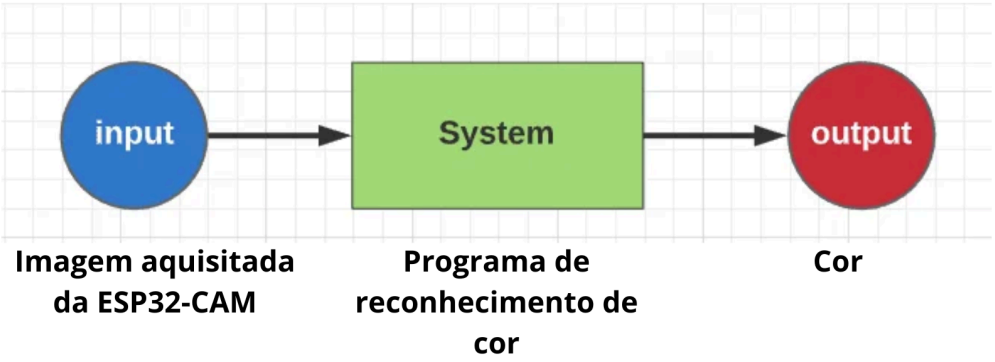
- Visual Studio Code (disponível em https://code.visualstudio.com/)
- Arduino IDE (opcional, disponível em https://www.arduino.cc/en/software)



Fluxograma de funcionamento do Kit Didático

O diagrama de funcionamento do software, mostrado na Figura acima, descreve o código de cada processador e a comunicação entre eles. Primeiramente, temos a ESP32, que controla o motor de passo e o sensor, responsáveis pela operação da esteira e da garra. Em paralelo, há uma câmera, representada pela ESP32-CAM, que realiza a aquisição de imagens e as envia continuamente para o servidor em um loop. O servidor, um PC, recebe as imagens da ESP32-CAM, analisa-as utilizando processamento digital de imagens e, após detectar o cubo, classifica sua cor. Em seguida, o servidor envia a informação da cor do cubo para a ESP32, que move a garra para colocá-lo na caixa correta.

A prática em questão será mais desenvolvida em cima da parte do Computador Pessoal, para isso é necessário que o código contemple as partes das comunicações



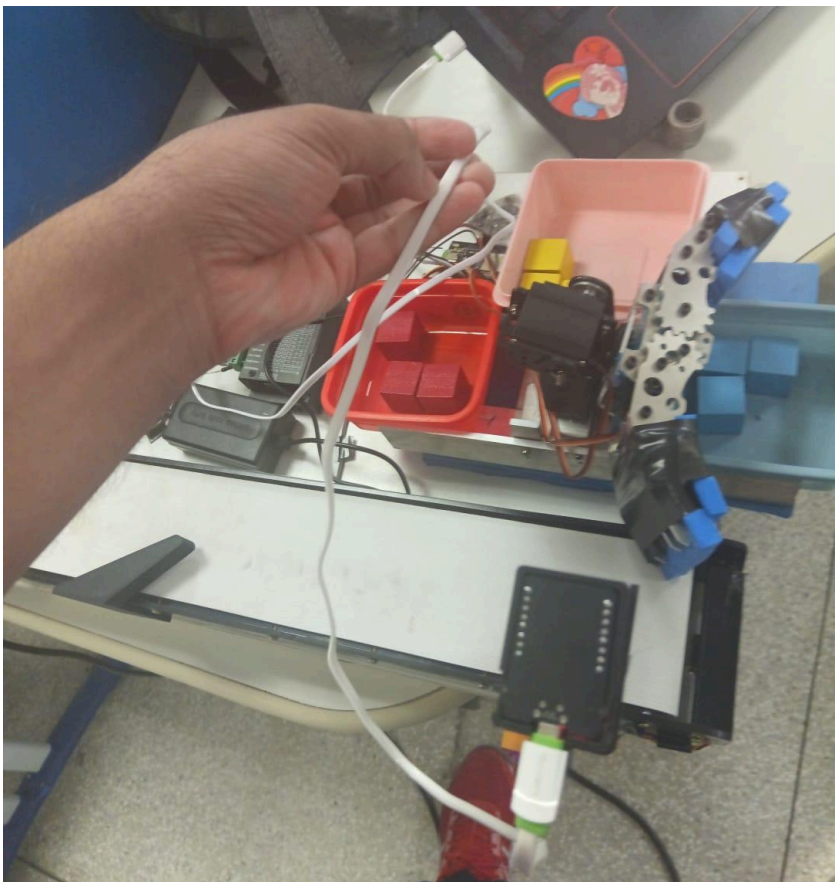
Será necessário que o código contemple a aquisição da imagem, proveniente da ESP32-CAM e a transmissão da cor para o ESP32.

Comunicação ESP32-CAM (socket)

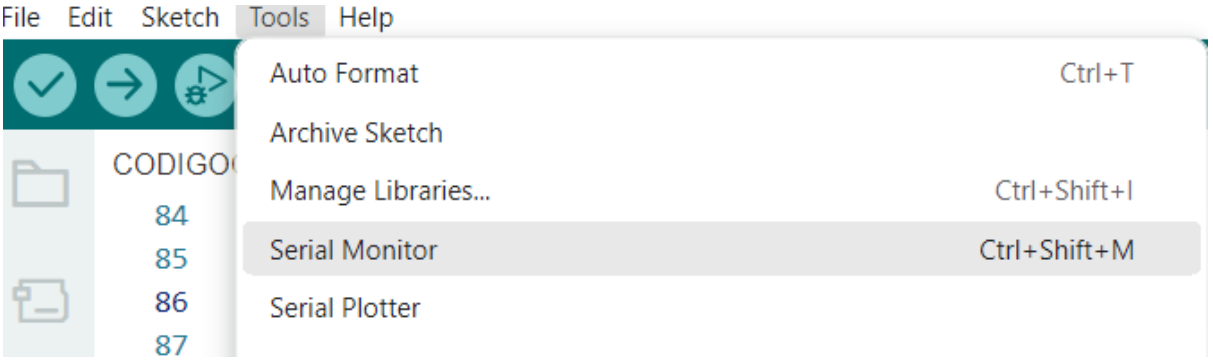
A aquisição da imagem é feita por essa linha de código:

```
# Replace the URL with the IP camera's stream URL
url = 'http://192.168.214.60/cam-hi.jpg'
```

Para saber esse endereço web basta plugar a ESP32-CAM diretamente ao PC.



Selecionar AI Thinker ESP32-CAM.



Abrir o Monitor Serial.

```
15:25:27.683 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
15:25:27.683 -> configsip: 0, SPIWP:0xee
15:25:27.715 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
15:25:27.715 -> mode:DIO, clock div:1
15:25:27.715 -> load:0x3fff0030,len:1344
15:25:27.715 -> load:0x40078000,len:13964
15:25:27.715 -> load:0x40080400,len:3600
15:25:27.715 -> entry 0x400805f0
15:25:28.227 ->
15:25:28.400 -> CAMERA OK
15:26:14.023 -> http://192.168.76.60
15:26:14.023 ->   /cam-lo.jpg
15:26:14.023 ->   /cam-hi.jpg
15:26:14.023 ->   /cam-mid.jpg
```

Emitirá essa mensagem contendo o endereço, complete com “/cam-hi.jpg” para obter a transmissão para seu programa em python.

É IMPORTANTE que a ESP32-CAM e o PC estejam no mesmo wifi. Para isso basta carregar o código da ESP32CAM que será passado ao final deste roteiro e modificar essas linhas de código em que WIFI_SSID é o nome da sua rede wifi e WIFI_PASS a senha da rede.

```
8   const char* WIFI_SSID = "NOTE 20 PRO";
9   const char* WIFI_PASS = "fluminense2012";
```

Comunicação ESP32 (serial)

A transmissão é feita por essas linhas de código:

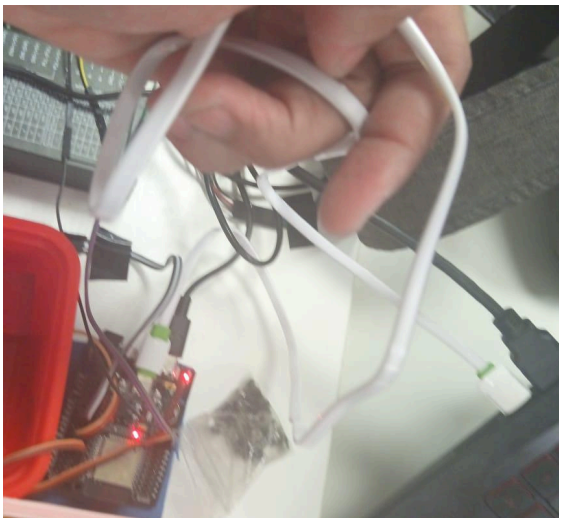
```
import serial
```

Inclui a biblioteca Serial.

```
# Configurar a comunicação serial
ser = serial.Serial('COM5', 9600)  # Altere 'COM3' para a porta serial correta
```

Configura a porta da comunicação para o programa em python, importante que seja a mesma presente na Arduino IDE.

Plugue o cabo da ESP32.



Aparecerá a respectiva COM endereçada no Arduino IDE.

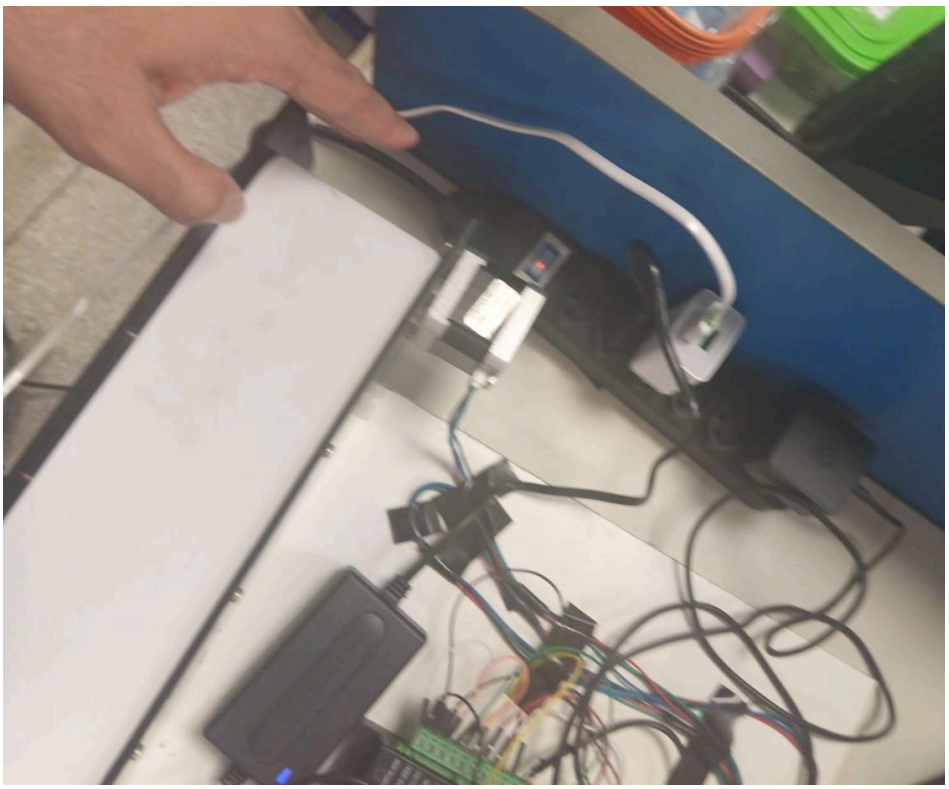
```
# Enviar a cor identificada para o ESP32
ser.write(cor_cubo.encode())
time.sleep(1)  # Pequena pausa para garantir que o dado seja enviado
```

Linhas de código para execução da transmissão da cor do cubo, a cor do cubo proveniente do código é a variável chamada “cor_cubo”.

Para ligar o kit plugue o filtro de linha em alguma tomada.

No filtro de linha estão ligados a ESP32-CAM, o motor de passo e o módulo expansor da ESP32.

Para ligar o Kit aperte o botão presente no filtro de linha, o motor de passo deve ligar no momento em que o botão for pressionado.



Fornecimento dos códigos:

Código ESP32-CAM:

```
#include <WebServer.h>
#include <WiFi.h>
#include <esp32cam.h>

#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"

const char* WIFI_SSID = "NOTE 20 PRO";
const char* WIFI_PASS = "fluminense2012";

WebServer server(80);


static auto loRes = esp32cam::Resolution::find(320, 240);
static auto midRes = esp32cam::Resolution::find(350, 530);
static auto hiRes = esp32cam::Resolution::find(800, 600);
```

```
void serveJpg()
{
  auto frame = esp32cam::capture();
  if (frame == nullptr) {
    Serial.println("CAPTURE FAIL");
    server.send(503, "", "");
    return;
  }
  Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(),
frame->getHeight(),
                static_cast<int>(frame->size()));

  server.setContentLength(frame->size());
  server.send(200, "image/jpeg");
  WiFiClient client = server.client();
  frame->writeTo(client);
}

void handleJpgLo()
{
  if (!esp32cam::Camera.changeResolution(loRes)) {
    Serial.println("SET-LO-RES FAIL");
  }
  serveJpg();
}

void handleJpgHi()
{
  if (!esp32cam::Camera.changeResolution(hiRes)) {
    Serial.println("SET-HI-RES FAIL");
  }
  serveJpg();
}

void handleJpgMid()
{
  if (!esp32cam::Camera.changeResolution(midRes)) {
    Serial.println("SET-MID-RES FAIL");
  }
  serveJpg();
}
```

```ino
void  setup(){
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
  Serial.begin(115200);
  Serial.println();
  {
    using namespace esp32cam;
    Config cfg;
    cfg.setPins(pins::AiThinker);
    cfg.setResolution(hiRes);
    cfg.setBufferCount(2);
    cfg.setJpeg(80);

    bool ok = Camera.begin(cfg);
    Serial.println(ok ? "CAMERA OK" : "CAMERA FAIL");
  }
  WiFi.persistent(false);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  Serial.print("http://");
  Serial.println(WiFi.localIP());
  Serial.println("  /cam-lo.jpg");
  Serial.println("  /cam-hi.jpg");
  Serial.println("  /cam-mid.jpg");

  server.on("/cam-lo.jpg", handleJpgLo);
  server.on("/cam-hi.jpg", handleJpgHi);
  server.on("/cam-mid.jpg", handleJpgMid);

  server.begin();
}

void loop()
{
  server.handleClient();
}
```

Código ESP32:

```
#include <Arduino.h>
#include <ESP32Servo.h>

Servo Garra, BaseGarra, Giro;  // cria um objeto Servo para a Garra

int pos = 0;    // variável para armazenar a posição do servo
// Pinos GPIO PWM recomendados no ESP32 incluem 2,4,12-19,21-23,25-27,32-33
int pinServoGarra = 12;
int pinServoGira = 5;
int pinServoBase = 19;

#define DIR_PIN  14  // Pino para controle da direção do motor
#define STEP_PIN 27  // Pino para controle do passo do motor
#define STEP_DELAY 820  // Ajuste esse valor para controlar a velocidade do
motor

const int sensorPin = 4; // Pino ao qual o sensor está conectado
int motorPin = 26; // Pino ao qual o motor está conectado

volatile bool detectado = false;
String corCubo = "";

void sensorTask(void * parameter) {
  (void)parameter;

  pinMode(sensorPin, INPUT_PULLUP); // Configura o pino do sensor como entrada
com pull-up interno

  for (;;) {
    int sensorValue = digitalRead(sensorPin); // Lê o valor do sensor
    if (sensorValue == LOW) { // Se o sensor detectar reflexão
      detectado = true;
      Serial.println("Detectado");
    } else {
      detectado = false;
      Serial.println("Não Detectado");
    }
    delay(100); // Pequena pausa para estabilidade
```

```
  }
}

void motorTask(void * parameter) {
  (void)parameter;

  pinMode(DIR_PIN, OUTPUT);
  pinMode(STEP_PIN, OUTPUT);
  pinMode(motorPin, OUTPUT); // Configura o pino do motor como saída

  for (;;) {
    if (!detectado) { // Se não houver detecção do sensor
      digitalWrite(DIR_PIN, LOW);  // Define a direção do motor (HIGH para uma direção, LOW para a outra)

      // Gira o motor continuamente em uma direção
      digitalWrite(STEP_PIN, HIGH);
      delayMicroseconds(STEP_DELAY);
      digitalWrite(STEP_PIN, LOW);
      delayMicroseconds(STEP_DELAY);
    } else {
      // Para o motor
      digitalWrite(STEP_PIN, LOW);

      // Executa a sequência da garra com base na cor detectada
      if (corCubo == "vermelho") {
        // Movimento para a direita
        Garra.write(32); delay(500);
        BaseGarra.write(100);  delay(1000);
        Giro.write(90); delay(2000);
        BaseGarra.write(19);  delay(2000);
        Garra.write(110); delay(500);
        BaseGarra.write(100);  delay(1000);
        Giro.write(0); delay(500);
        BaseGarra.write(19);  delay(1000);
        Garra.write(32);
      } else if (corCubo == "amarelo") {
        // Movimento para o centro
        Garra.write(32); delay(500);
        BaseGarra.write(100);  delay(1000);
        Giro.write(90); delay(2000);
        BaseGarra.write(19);  delay(2000);
```

```
        Garra.write(110); delay(500);
        BaseGarra.write(180);  delay(1000);
        Garra.write(32);
      } else if (corCubo == "azul") {
        // Movimento para a esquerda
        Garra.write(32); delay(500);
        BaseGarra.write(100);  delay(1000);
        Giro.write(90); delay(2000);
        BaseGarra.write(19);  delay(2000);
        Garra.write(110); delay(500);
        BaseGarra.write(100);  delay(1000);
        Giro.write(180); delay(500);
        BaseGarra.write(19);  delay(1000);
        Garra.write(32);
      }

      detectado = false; // Reseta a variável de detecção
    }

    if (Serial.available() > 0) {
      corCubo = Serial.readString();
      Serial.println("Cor recebida: " + corCubo);
    }
  }
}

void setup() {
  Serial.begin(9600);

  Garra.attach(pinServoGarra); // Anexa o servo ao pino correspondente
  BaseGarra.attach(pinServoBase); // Anexa o servo ao pino correspondente
  Giro.attach(pinServoGira);

  xTaskCreatePinnedToCore(sensorTask, "Sensor Task", 10000, NULL, 1, NULL, 0);
  xTaskCreatePinnedToCore(motorTask, "Motor Task", 10000, NULL, 1, NULL, 1);
}

void loop() {
  // nothing to do here
}
```

Esqueleto para Código VSCode:

```
import cv2
import numpy as np
import urllib.request
import serial
import time

# Configurar a comunicação serial
ser = serial.Serial('COM5', 9600)  # Altere 'COM3' para a porta serial correta

# Função para identificar a cor

# Replace the URL with the IP camera's stream URL
url = 'http://192.168.76.60/cam-hi.jpg'

# Create a VideoCapture object
cap = cv2.VideoCapture(url)

# Check if the IP camera stream is opened successfully
if not cap.isOpened():
    print("Failed to open the IP camera stream")
    exit()

# Read and display video frames
while True:
    # Read a frame from the video stream
    img_resp=urllib.request.urlopen(url)
    imgnp=np.array(bytearray(img_resp.read()),dtype=np.uint8)
    im = cv2.imdecode(imgnp,-1)


    # Identificar a cor do cubo na zona de interesse
    cor_cubo = identificar_cor_zona_interesse(imagem_saturacao_aumentada,
zona_interesse)

        # Enviar a cor identificada para o ESP32
        ser.write(cor_cubo.encode())
        time.sleep(1)  # Pequena pausa para garantir que o dado seja enviado
```

```
    else:
        print("Nenhuma cor foi identificada.")

    cv2.imshow('Identificacao de Cubo', im)
    key=cv2.waitKey(5)
    if key==ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
ser.close()
```