

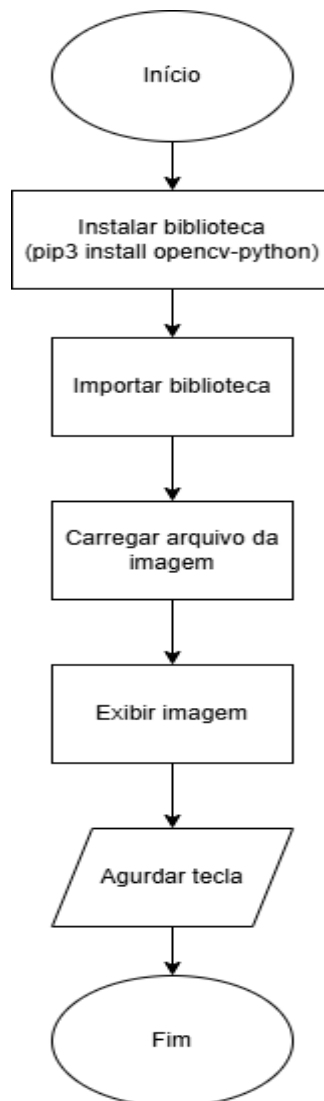
# Relatório de progresso: Reconhecimento de Display 7 segmentos com OpenCV

06/06/2025

## Visão geral

Instalação da biblioteca e primeiros passos para leitura do arquivo e exibição de imagem em linguagem Python.

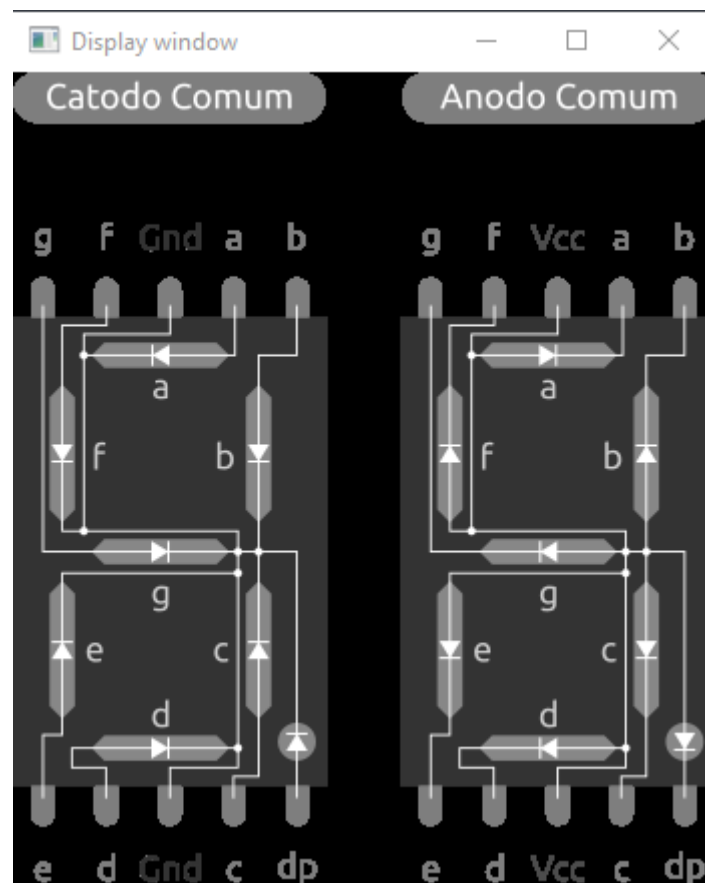
## Fluxograma



## Código

```
1 import cv2 as cv
2
3 img = cv.imread("img/display.png", cv.IMREAD_GRAYSCALE)
4 cv.imshow("Display window", img)
5 k = cv.waitKey(0)
```

## Saída



**24/06/2025**

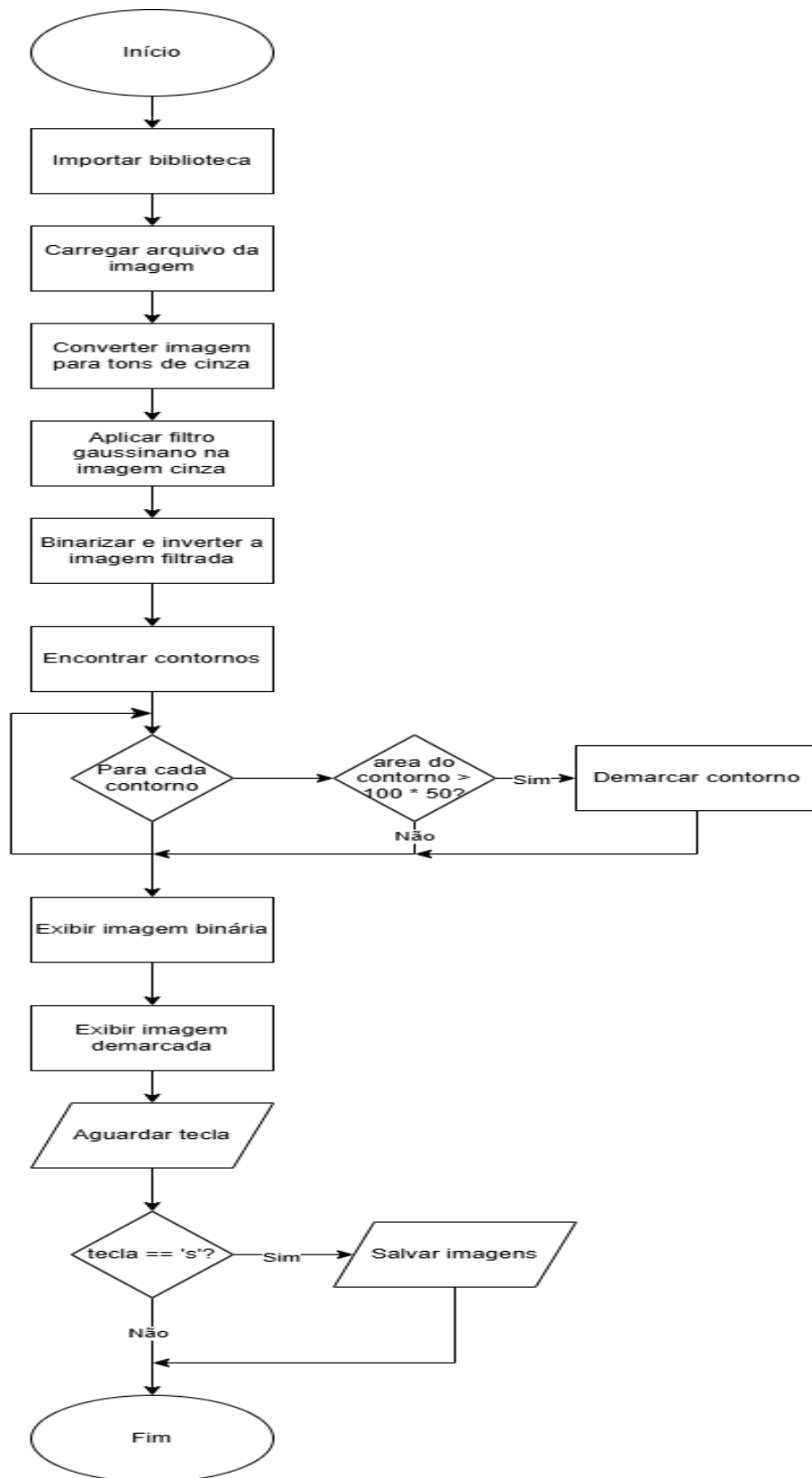
## **Visão Geral**

Testes básicos de aplicação de filtros de cor e binarização da imagem para extração de dados.

Uma imagem amostral de um multímetro foi escolhida para a realização do procedimento. As etapas foram:

- Converter a imagem para tons de cinza
- Aplicar filtro gaussiano para “borrar” levemente a imagem e reduzir ruídos nas bordas
- Binarizar a imagem apenas para tons pixels pretos ou brancos utilizando um valor limiar de intensidade
- Inverter a imagem binária
- Encontrar contornos utilizando os métodos internos do OpenCV
- Filtrar os contornos encontrados (carece de ajuste)
- Demarcar contornos

## Fluxograma



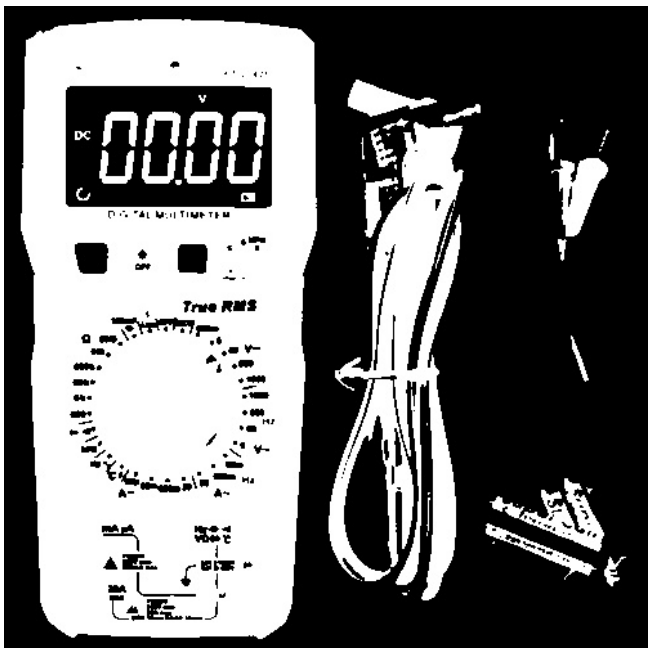
## Código

```
1 import cv2 as cv
2
3 # Carregar e redimensionar imagem
4 img_dir = r'..\\img'
5 img = cv.imread(f'{img_dir}\\multimetro.jpg')
6 img = cv.resize(img, (500, 500))
7
8 # Converter imagem para preto e branco e aplicar filtro gaussiano
9 gray_img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
10 blurred_img = cv.GaussianBlur(gray_img, (3, 3), 0)
11
12 # Binarizar imagem
13 _, thresh = cv.threshold(blurred_img, 120, 255, cv.THRESH_BINARY_INV)
14
15 # Encontrar contornos
16 contours, _ = cv.findContours(thresh, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)
17
18 # Filtrar contornos através de uma área mínima para evitar pequenos ruídos (carece de ajustes)
19 for cnt in contours:
20     area = cv.contourArea(cnt)
21     if area > 100 * 50:
22         x, y, w, h = cv.boundingRect(cnt)
23         cv.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
24
25 cv.imshow("Imagem binarizada", thresh)
26 cv.imshow("Imagem com contorno", img)
27 k = cv.waitKey(0)
28
29 if chr(k) == 's':
30     cv.imwrite(f'{img_dir}\\multimetro_binario.jpg', thresh)
31     cv.imwrite(f'{img_dir}\\multimetro_demarcado.jpg', img)
```

Entrada



Saída



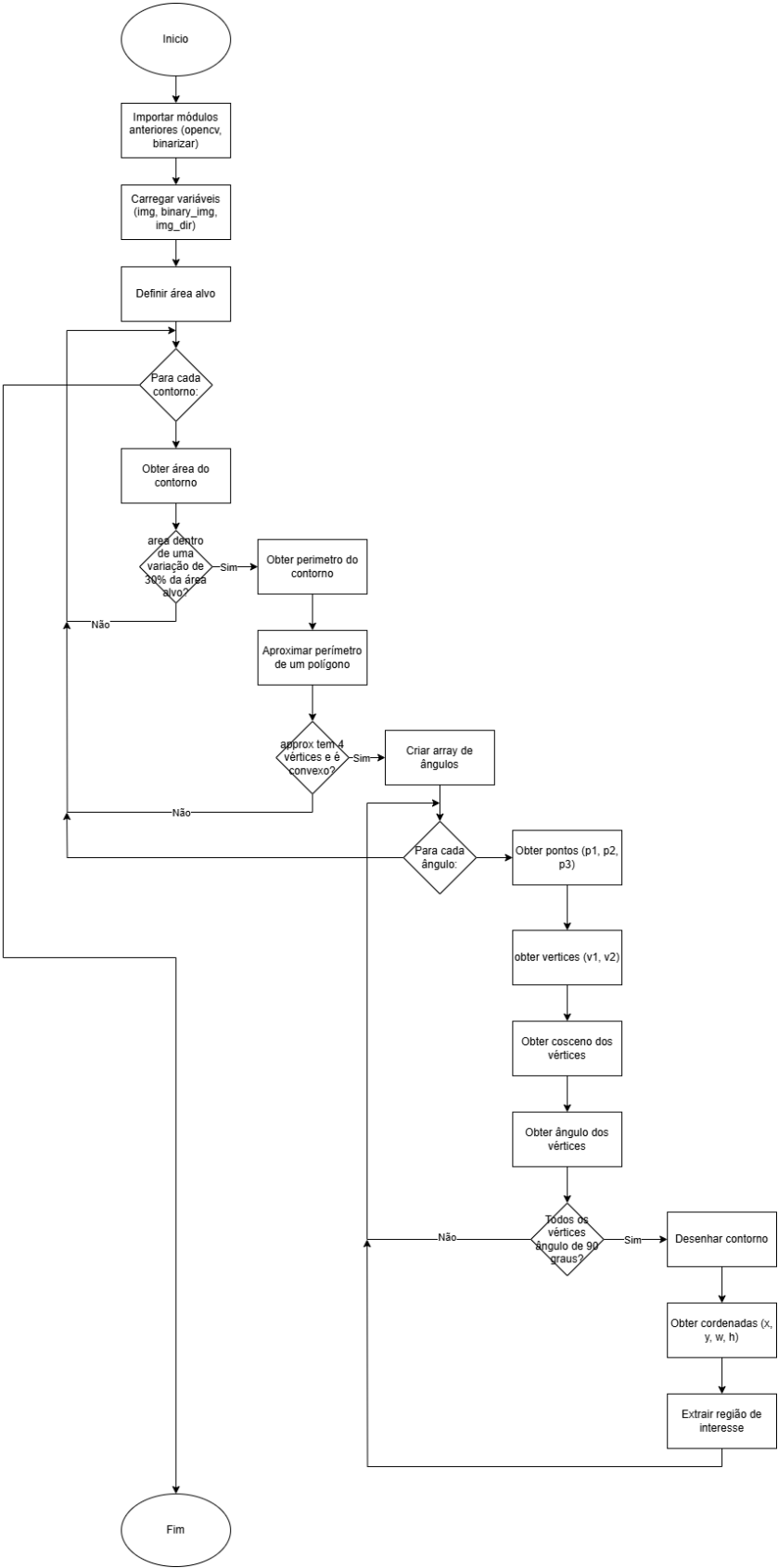


28/06/2025 (1)

## Visão geral

Obtenção da região de interesse (ROI) do display através de algoritmos de aproximação em cima do perímetro do contorno e refinamento da área de filtro.

# Fluxograma





## Código

```
1  import cv2 as cv
2  import numpy as np
3  import binarizar
4
5  img_dir = binarizar.img_dir
6  img = binarizar.img
7  binary_img = binarizar.thresh
8
9  # Filtrar contornos através de uma área mínima para evitar pequenos ruídos
10 target_area = 162 * 97 # Número literal. Valores mais adequados precisam ser encontrados.
11 for cnt in binarizar.contours:
12     area = cv.contourArea(cnt)
13     # Verifica se a área do contorno está dentro das tolerâncias
14     if area > target_area * 0.7 and area < target_area * 1.3:
15         # Tenta aproximar o perímetro do contorno a um retângulo
16         perimeter = cv.arcLength(cnt, True)
17         approx = cv.approxPolyDP(cnt, 0.02 * perimeter, True)
18
19         # Verifica se o contorno aproximado possui
20         if len(approx) == 4 and cv.isContourConvex(approx):
21             angles = []
22             for i in range(4):
23                 p1 = approx[i][0]
24                 p2 = approx[(i + 1) % 4][0]
25                 p3 = approx[(i + 2) % 4][0]
26
27                 v1 = p1 - p2
28                 v2 = p3 - p2
29                 cos_angle = np.dot(v1, v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))
30                 angle = np.degrees(np.arccos(cos_angle))
31                 angles.append(angle)
32
33             if all(a >= 85 and a <= 95 for a in angles):
34                 cv.drawContours(img, [cnt], 0, (0, 255, 0), 3)
35                 x, y, w, h = cv.boundingRect(cnt)
36                 roi = img[y:y+h, x:x+w]
37
38 cv.imshow("Imagem binarizada", binary_img)
39 cv.imshow("Imagem com contorno", img)
40 cv.imshow("Display", roi)
41 k = cv.waitKey(0)
42
43 if chr(k) == 's':
44     contour_path = f'{img_dir}\\multimetro_demarcado.jpg'
45     display_path = f'{img_dir}\\display.jpg'
46     cv.imwrite(contour_path, img)
47     cv.imwrite(display_path, roi)
48     print(f'Imagem guardada em: {contour_path}'.replace('.\\', ''))
49     print(f'Imagem guardada em: {display_path}'.replace('.\\', ''))
```

Entrada



Saída



28/06/2025 (2)

## Visão geral

Reorganização das rotinas de pré-processamento e tratamento de imagem em funções para reutilização e testes com diversas imagens amostrais. O processo foi reorganizado para isolar o aparelho (um multímetro, neste caso) e depois identificar o display.

## Entradas





















