

# PROCESAMIENTO DIGITAL DE IMÁGENES I

Trabajo Práctico N°2

Año 2024 - Primer Semestre

**Grupo 8:**

Cancio José

García Julián

Herrera Francisca

[INTRODUCCIÓN](#)

[PROBLEMA 1](#)

[Detección y clasificación de componentes electrónicos en PCB](#)

[Introducción](#)

[Comentarios sobre la resolución](#)

[Aislamiento de componentes](#)

[Ejercicio A: Generar imagen de salida con segmentación](#)

[Ejercicio B: Clasificación de capacitores](#)

[Ejercicio C: Contar resistencias](#)

[PROBLEMA 2](#)

[Detección y segmentación de patentes y caracteres](#)

[Enunciado](#)

[Comentarios sobre la resolución](#)

[Conclusiones](#)

[PROBLEMA 1](#)

[PROBLEMA 2](#)

## INTRODUCCIÓN

Este trabajo práctico consiste en la resolución de dos problemas relacionados con el procesamiento de imágenes.

# PROBLEMA 1

## Detección y clasificación de componentes electrónicos en PCB

### Introducción

El objetivo de este informe es detallar la resolución del problema de detección y clasificación de componentes electrónicos en una imagen de una placa de circuito impreso (PCB). La imagen proporcionada contiene una variedad de componentes electrónicos. Las tareas a desarrollar incluyen segmentar y distinguir tres tipos principales de componentes (resistencias eléctricas, capacitores y el chip), clasificar los capacitores según su tamaño y contar la cantidad de resistencias presentes en la imagen.

### Comentarios sobre la resolución

Para resolver este problema, se han empleado diversas técnicas de procesamiento de imágenes y clustering. Las principales herramientas y bibliotecas utilizadas incluyen OpenCV para el procesamiento de imágenes, matplotlib para la visualización y scikit-learn para el clustering.

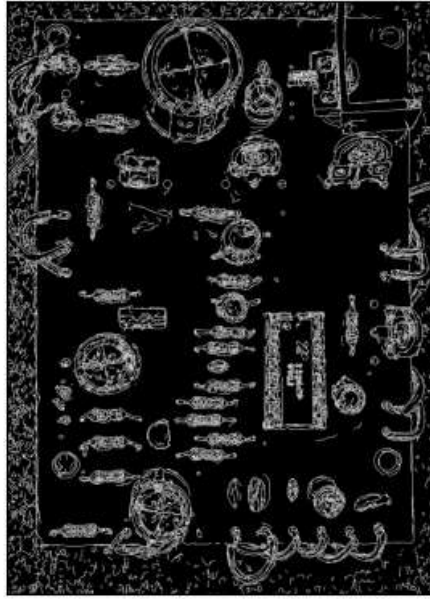
#### Funciones Auxiliares

- + *imshow*: Para mostrar imágenes con distintas configuraciones.
- + *erosion* y *dilation*: Para aplicar operaciones morfológicas de erosión y dilatación.
- + *filtro\_area*: Para filtrar los contornos de acuerdo a un umbral de área.
- + *filtro\_forma*: Para filtrar contornos basados en la circularidad.
- + *label\_connected\_components*: Para etiquetar y visualizar componentes conectados.

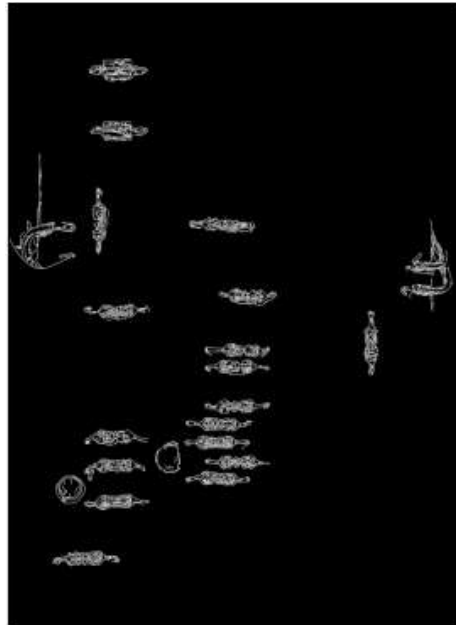
### Aislamiento de componentes

#### 1. Aislamiento de resistencias

- Se carga la imagen en formato RGB y se aplica un filtro de mediana para reducir el ruido.
- La imagen se convierte a escala de grises y se aplica el detector de bordes de Canny para resaltar los bordes de los componentes.



- Aplicación de detección de bordes, dilatación y filtrado basado en área y forma para aislar las resistencias.

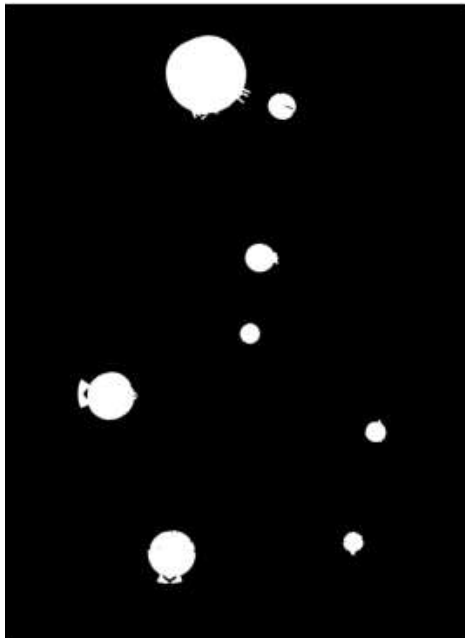


- Uso de operaciones morfológicas para refinar la segmentación.



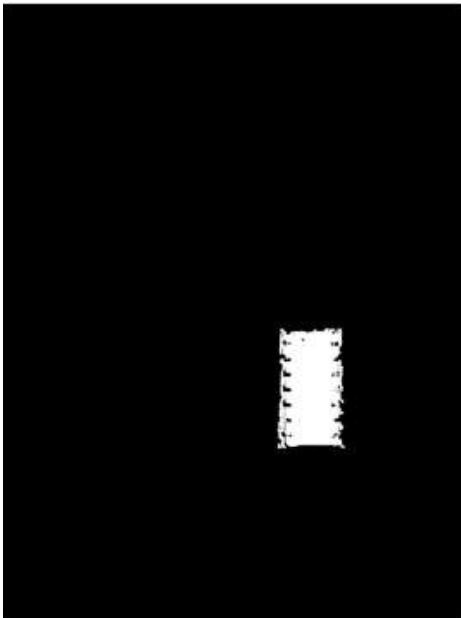
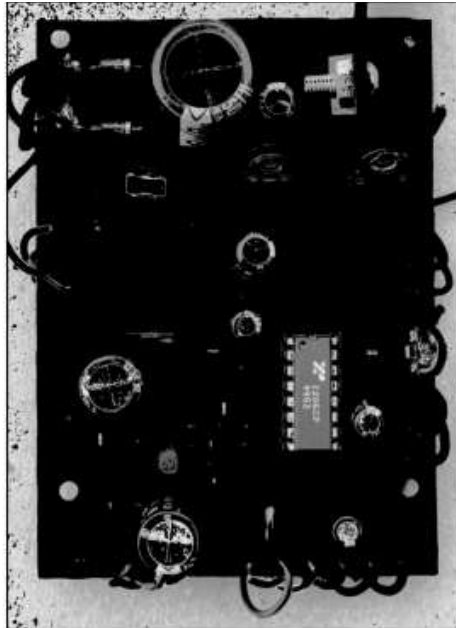
## 2. Aislamiento de capacitores

- Conversión de la imagen a escala de grises y aumento de brillo para resaltar capacitores.
- Umbralización y filtrado basado en área y forma para aislar los capacitores.



### 3. Aislamiento del chip

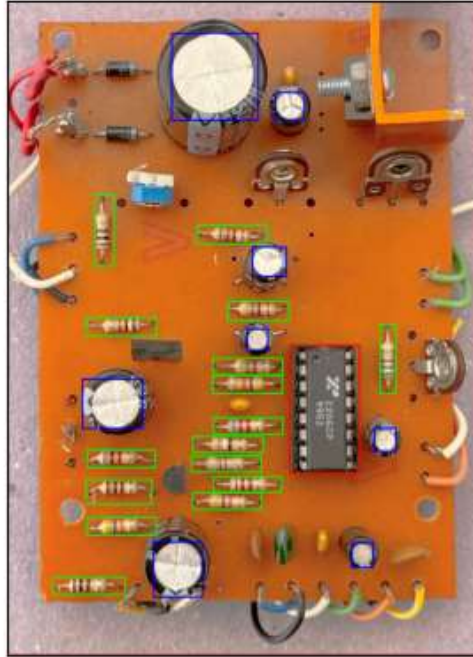
- Conversión de la imagen a espacio de color HSV para segmentar el chip basado en sus tonos de gris.
- Aplicación de máscara y operaciones morfológicas para refinar la segmentación.



### **Ejercicio A: Generar imagen de salida con segmentación**

Para cada tipo de componente segmentado, se dibujan rectángulos alrededor de sus contornos en la imagen original:

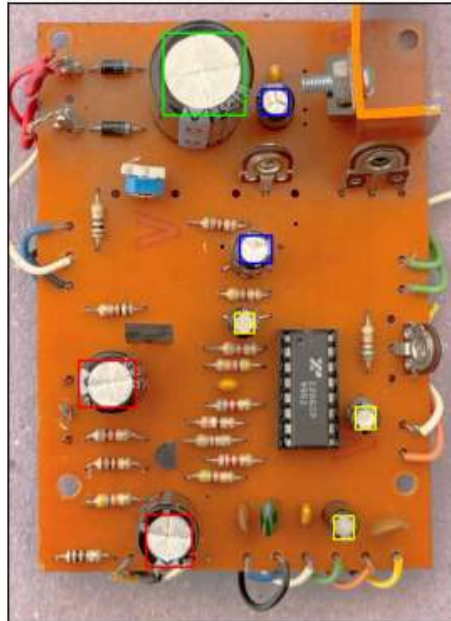
- Resistencias: Verde
- Capacitores: Azul
- Chip: Rojo



### Ejercicio B: Clasificación de capacitores

- Se utiliza K-means clustering para agrupar los capacitores según su tamaño.
- Se dibujan rectángulos de diferentes colores alrededor de los capacitores según su grupo y se cuenta la cantidad de capacitores en cada categoría

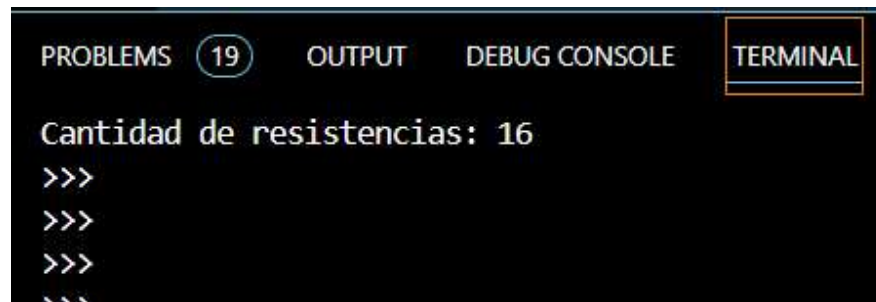




```
PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL
...
Cantidad de capacitores de color rojo: 2
Cantidad de capacitores de color verde: 1
Cantidad de capacitores de color azul: 2
Cantidad de capacitores de color amarillo: 3
>>>
<<<
```

## Ejercicio C: Contar resistencias

- Se etiqueta cada resistencia segmentada y se cuenta el número total de resistencias presentes en la imagen.



```
PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL
Cantidad de resistencias: 16
>>>
>>>
>>>
>>>
```

## PROBLEMA 2

### Detección y segmentación de patentes y caracteres

#### Introducción

El objetivo de este informe es detallar la resolución del problema de detección automática y segmentación de patentes y caracteres que contienen las mismas proporcionadas por varias imágenes. Las tareas a desarrollar incluyen segmentar y distinguir las placas patentes de cada automóvil para luego segmentar los caracteres que las conforman.

#### Comentarios sobre la resolución

Para resolver este problema, se han empleado diversas técnicas de procesamiento de imágenes. Las principales herramientas y bibliotecas utilizadas incluyen OpenCV para el procesamiento de imágenes, matplotlib para la visualización.

#### Funciones Auxiliares

- + *imshow*: Para mostrar imágenes con distintas configuraciones.
- + *dilation*: Para aplicar operaciones morfológicas de dilatación.
- + *filtro\_area*: Para filtrar los contornos de acuerdo a un umbral de área.
- + *filtro\_por\_relacion\_aspecto*: Para filtrar contornos basados en la relación de aspecto ancho/alto de la caja contenedora.
- + *patente*: Recibe la imagen original, un valor p binario para impresión y devuelve:
  - + la imagen original con la segmentación de la patente.
  - + la matriz stats de los componentes conectados (el fondo y la patente).
- + *crop\_patente*: Recibe la imagen original, un valor p binario para impresión y devuelve la imagen recortada de la patente. Llama a la función *patente*.
- + *comp\_patente*: Recibe la imagen original, un valor de umbral, un valor p binario para impresión y devuelve la imagen recortada de la patente con la segmentación de las letras. Llama a la función *crop\_patente*.

- + *letras*: Recibe la imagen original y llama a la función `comp_patente`, iterando sobre el umbral de binarización y el mejoramiento de nitidez de la imagen, ajustando estos parámetros para maximizar la cantidad de componentes conectados hasta el valor 7 (el fondo de la imagen y los seis caracteres que identifican a la patente). Devuelve la imagen recortada de la patente con la segmentación de las letras.
- + *detector*: Recibe como parámetro el nombre del archivo que contiene la imagen y llama consecutivamente a las funciones *letras* y *patente*, devolviendo:
  - + la imagen original con la segmentación de la patente.
  - + la imagen recortada de la patente con la segmentación de las letras.

## 1. Detallado de la función *patente*

- Se carga la imagen original, se pasa a grises y aumenta el contraste. Se aumenta el brillo y umbraliza. Hay un mejoramiento de la nitidez que se aplica según el caso.



Umbralado



- Se filtran los componentes conectados sucesivamente por relación de aspecto y áreas para encontrar el componente conectado que coincida con la patente.

Filtrado Por Relacion65



Filtrado Por Area 300



Dilatacion y Filtrado Por Area500



Filtrado Por Area700



- Se segmenta la patente sobre la imagen original, y se devuelve la matriz stats de los componentes conectados.

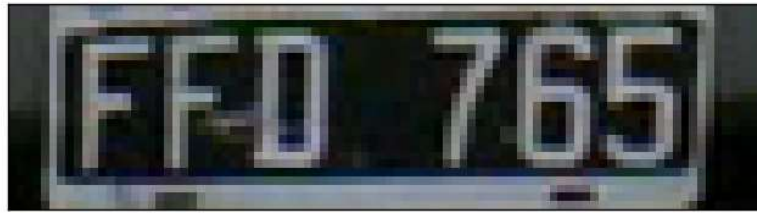


- Se pudieron detectar 10 de las 12 patentes (no se encuentran en las imágenes 05 y 07).

## 2. Detallado de la función crop\_patente

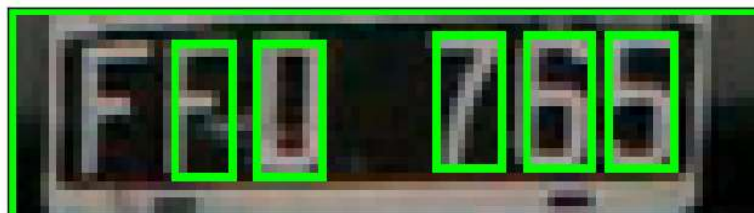
- Recibe la salida de la función patente y recorta la imagen a partir de la posición y dimensiones de la caja contenedora del componente de la patente.

Cropped Image



### 3. Detallado de la función comp\_patente

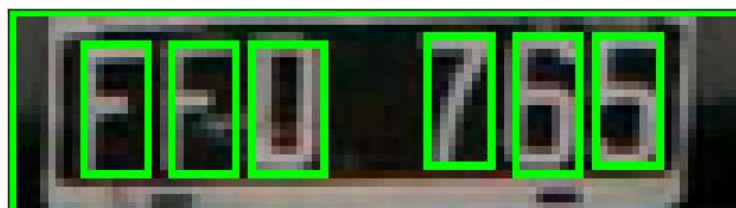
- Recibe la salida de la función crop\_patente, binariza la imagen de la patente recortada y realiza sucesivos filtros de área y relación de aspecto para luego segmentar las letras.



- La cantidad de letras detectadas varía según el valor de umbral utilizado en la binarización. Se probaron distintos umbrales. Entre 110 y 150, con un salto de 10, se pudieron detectar los 6 caracteres de las 10 patentes. Se continúa el desarrollo en la siguiente función.

### 4. Detallado de la función letras

- Recibe la salida de la función comp\_patente e itera para distintos valores de umbral. Así se obtiene:



- Para las imágenes 05 y 07 se resuelve el problema de la detección de las patentes mejorando la nitidez de la imagen en la función patente. El mejoramiento de la nitidez afecta el resultado de otras imágenes, por ende se analizan ambas opciones (kernel de nitidez True o False) para obtener el

resultado que maximice el número de letras obtenidas (hasta 6). Así se logra resolver la detección para el 100% de las patentes y caracteres.

## Conclusiones

### PROBLEMA 1

El código desarrollado realiza eficazmente la segmentación y clasificación de los componentes electrónicos en la imagen de la PCB. Las técnicas de procesamiento de imágenes y clustering utilizadas permiten una identificación y visualización clara de los componentes, así como la clasificación precisa de los capacitores y el conteo de resistencias. Este enfoque puede ser aplicado a otras tareas similares de análisis de imágenes en el ámbito de la ingeniería y la electrónica.

### PROBLEMA 2

El código desarrollado realiza la detección de la placa patente y la segmentación de los caracteres que identifican a la patente. Las técnicas de procesamiento de imágenes utilizadas permiten una identificación y visualización clara. Fue interesante buscar alternativas cada vez que se presentaron resultados distintos a los deseados. No obstante, se podría seguir trabajando en una detección más precisa de la patente, tal vez rotarla para que los caracteres no queden como en perspectiva, aplicar reconocimiento óptico de caracteres para extraerlos y almacenarlos, etc.