In [2]:
```
import pytesseract
from PIL import Image

import cv2
from matplotlib import pyplot as plt
```

In [3]:
```
def display(im_path):
    dpi = 80
    im_data = plt.imread(im_path)

    height, width  = im_data.shape[:2]

    # What size does the figure need to be in inches to fit the image?
    figsize = width / float(dpi), height / float(dpi)

    # Create a figure of the right size with one axes that takes up the full figure
    fig = plt.figure(figsize=figsize)
    ax = fig.add_axes([0, 0, 1, 1])

    # Hide spines, ticks, etc.
    ax.axis('off')

    # Display the image.
    ax.imshow(im_data, cmap='gray')

    plt.show()
```

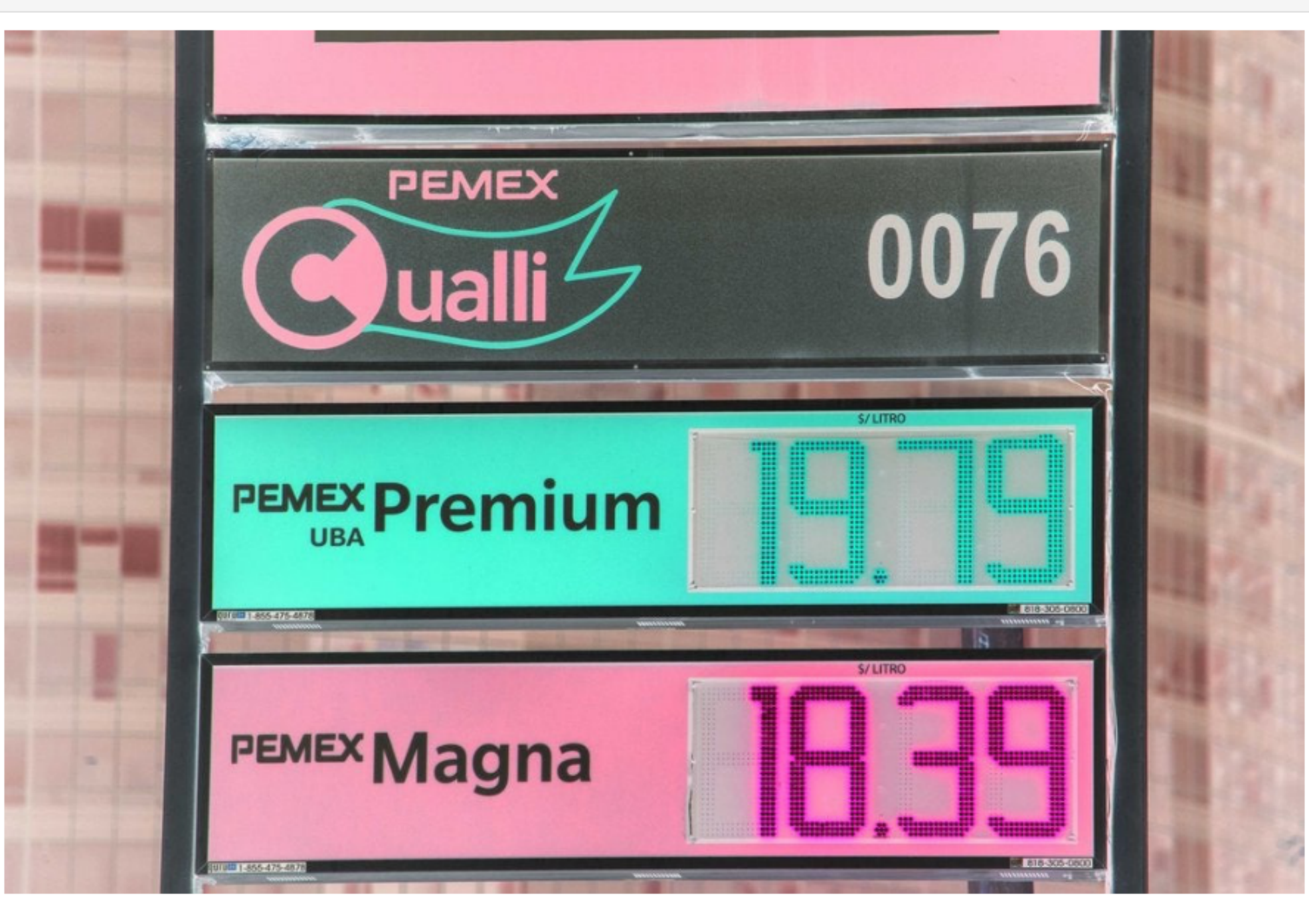
Primero cargamos la imagen de la ubicación en dónde está:

In [4]:
```
image_file = '/home/jos/Desktop/scriptsJos/ImagenesGaspre/2.jpg'
img1 = Image.open(image_file)
img = cv2.imread(image_file)
display(image_file)
```

Comenzamos el proceso de preprocesamiento de la imagen, probando diferentes métodos para obtener los mejores resultados de lectura.

Primero invertimos los colores de la imagen, que ayuda en casos en que se tiene texto claro en un fondo obscuro, como es el caso del precio de Magna en la imagen anterior.

In [5]:
```
inverted_image = cv2.bitwise_not(img)
cv2.imwrite("/home/jos/Desktop/scriptsJos/ImagenesGaspre/inverted.jpg", inverted_image)
display("/home/jos/Desktop/scriptsJos/ImagenesGaspre/inverted.jpg")
```

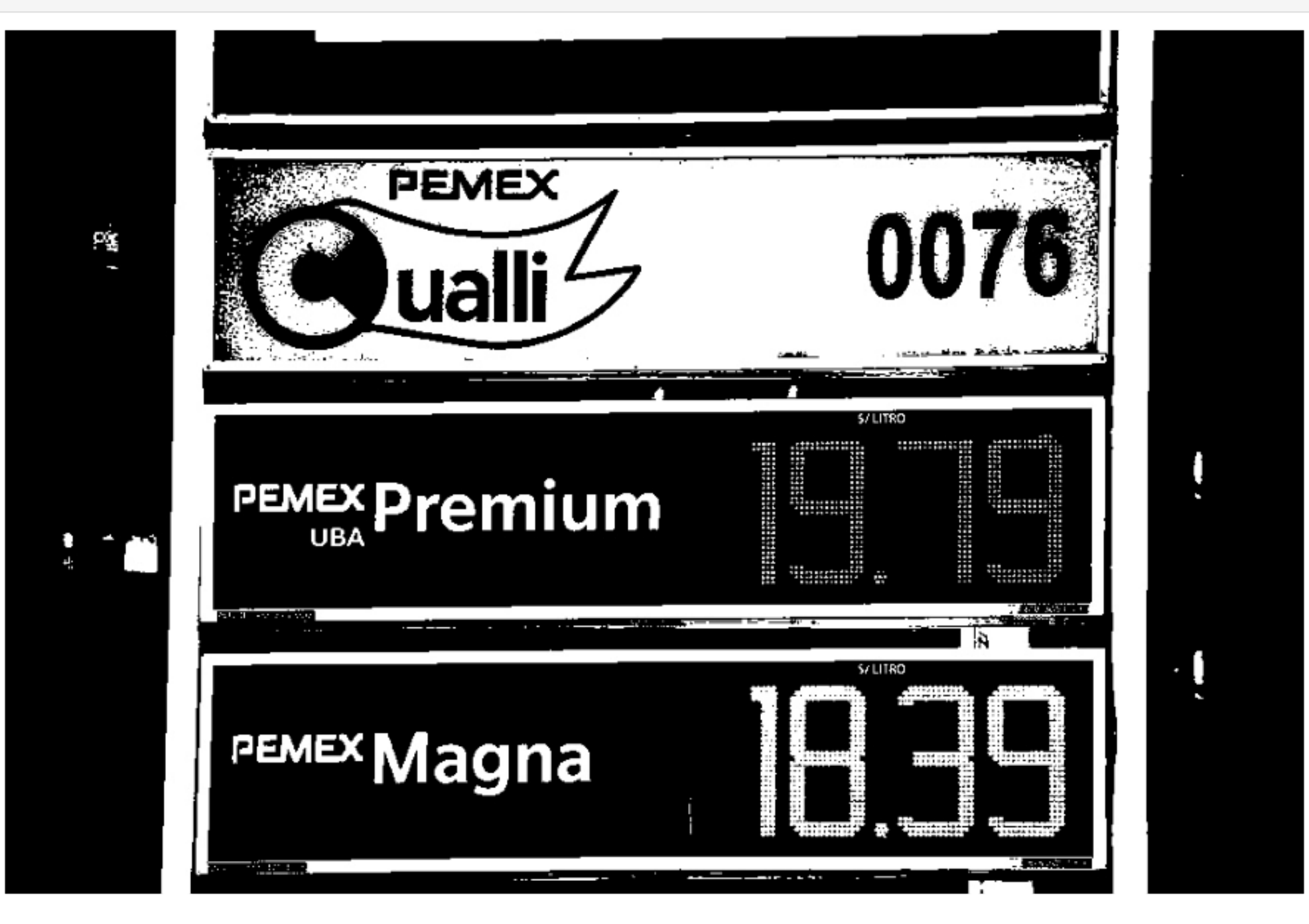


Ahora pasamos la imagen a una escala de grises, paso que es necesario para la binarización de la imagen.

In [6]:
```
def grayscale(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
gray_image = grayscale(img)
cv2.imwrite("/home/jos/Desktop/scriptsJos/ImagenesGaspre/gray.jpg", gray_image)
display("/home/jos/Desktop/scriptsJos/ImagenesGaspre/gray.jpg")
```

Ya que se tiene la imagen en una escala de grises, escogemos parámetros para elegir que pasará a blanco y que pasará a negro, en caso de que no se obtengan buenos resultados con los primeros parámetros establecidos se reajustan los parámetros y se vuelve a realizar el proceso.

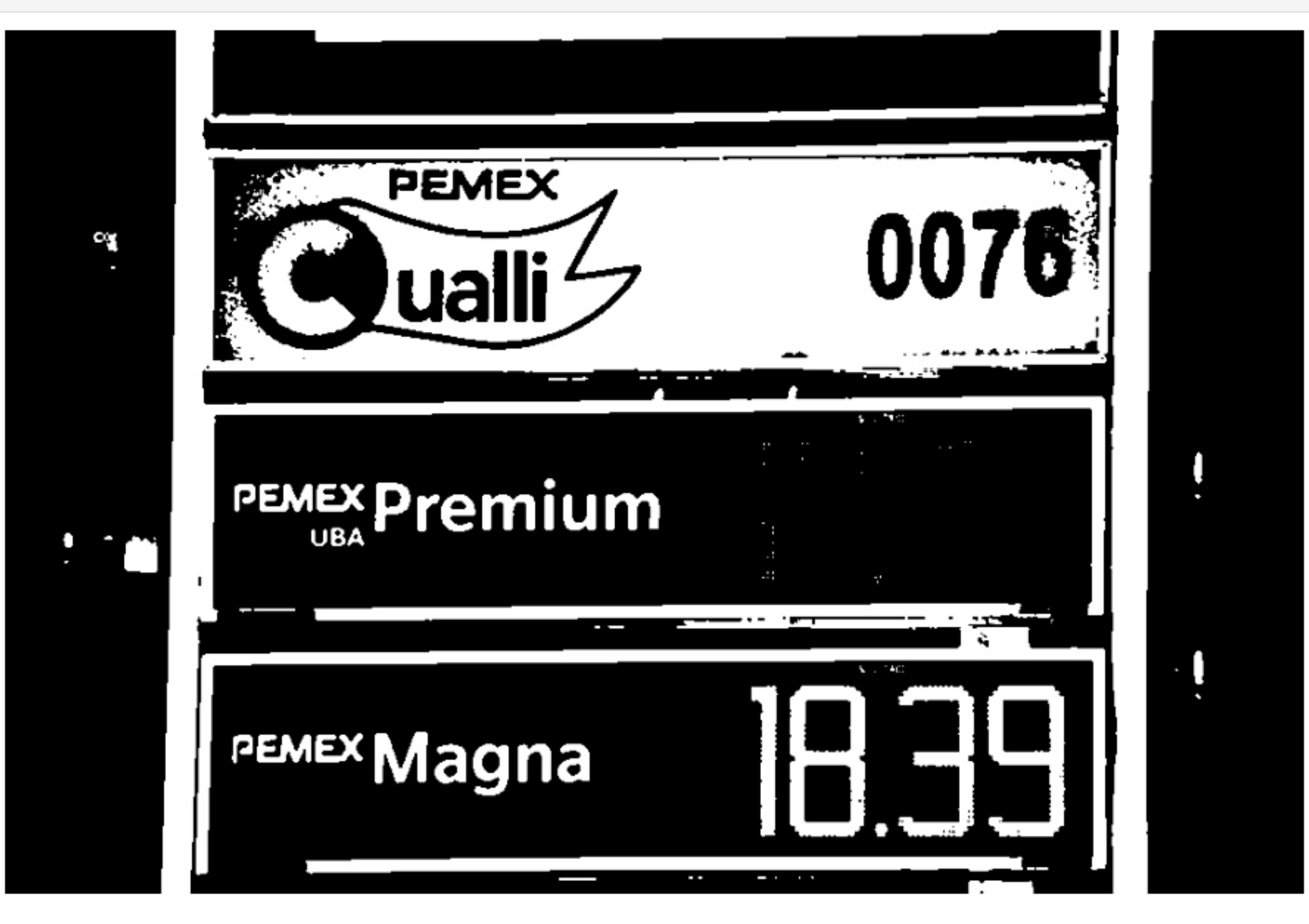
In [7]:
```
thresh, im_bw = cv2.threshold(grayscale(img), 150, 280, cv2.THRESH_BINARY)
cv2.imwrite("/home/jos/Desktop/scriptsJos/ImagenesGaspre/bw_image.jpg", im_bw)
display("/home/jos/Desktop/scriptsJos/ImagenesGaspre/bw_image.jpg")
```



Después se realiza un paso de reducción de ruido para remover sombras que haya sobre el texto o puntos de luz sobre el fondo. En caso de que se tome como ruido como es el caso del precio de Magna agregamos un paso de hacer más grueso el texto.

In [8]:
```
def noise_removal(image):
    import numpy as np
    kernel = np.ones((1, 1), np.uint8)
    image = cv2.dilate(image, kernel, iterations=1)
    kernel = np.ones((1, 1), np.uint8)
    image = cv2.erode(image, kernel, iterations=1)
    image = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernel)
    image = cv2.medianBlur(image, 3)
    return (image)
```

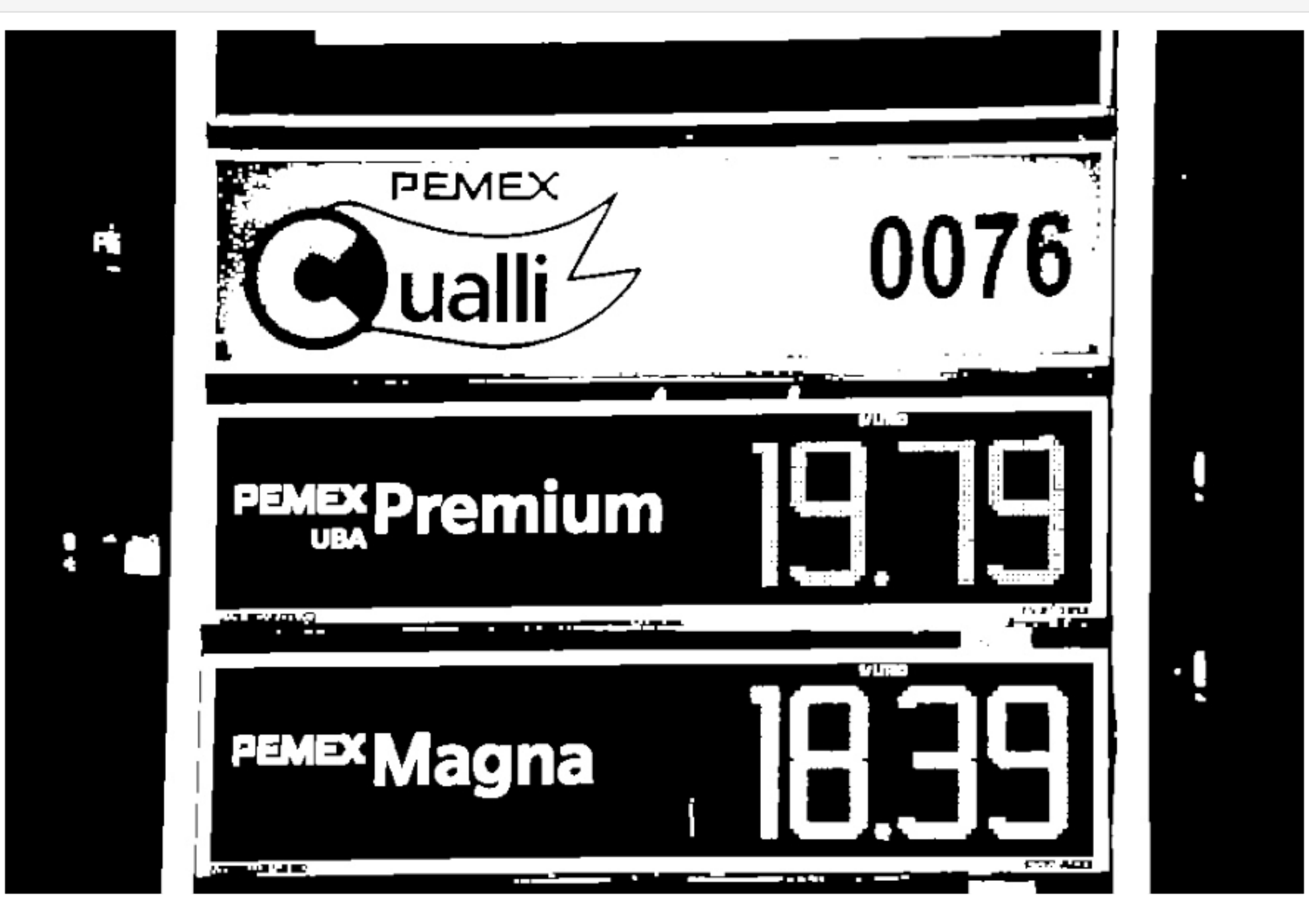
In [9]:
```
no_noise = noise_removal(im_bw)
cv2.imwrite("/home/jos/Desktop/scriptsJos/ImagenesGaspre/no_noise.jpg", no_noise)
display("/home/jos/Desktop/scriptsJos/ImagenesGaspre/no_noise.jpg")
```



Hacemos paso de hacer más grueso el texto.

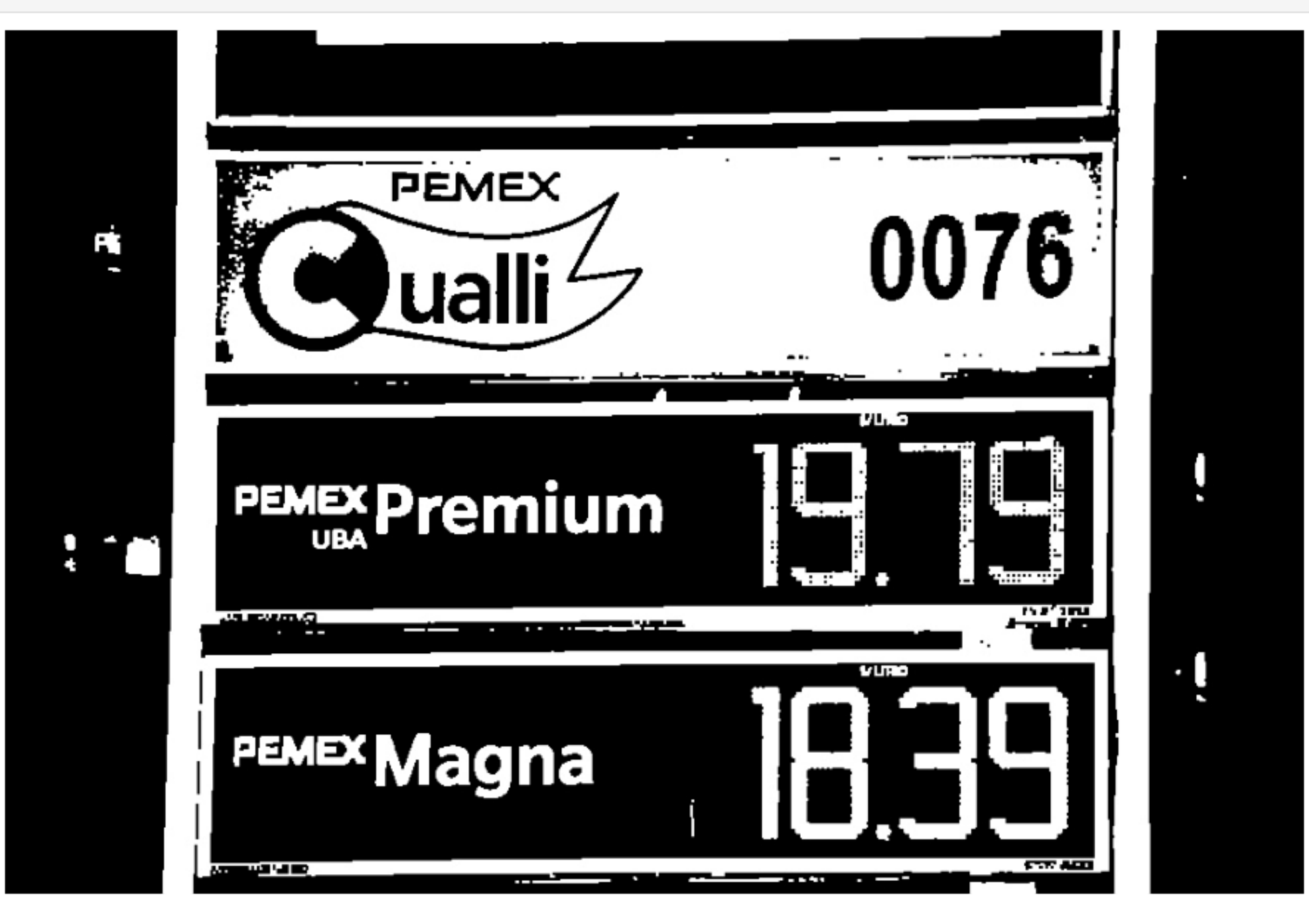
In [10]:
```
def thin_font(image):
    import numpy as np
    image = cv2.bitwise_not(image)
    kernel = np.ones((2,2),np.uint8)
    image = cv2.erode(image, kernel, iterations=2)
    image = cv2.bitwise_not(image)
    return (image)
```

In [11]:
```
eroded_image = thin_font(im_bw)
cv2.imwrite("/home/jos/Desktop/scriptsJos/ImagenesGaspre/eroded_image.jpg", eroded_image)
display("/home/jos/Desktop/scriptsJos/ImagenesGaspre/eroded_image.jpg")
```



In [12]:
```
def thick_font(image):
    import numpy as np
    image = cv2.bitwise_not(image)
    kernel = np.ones((2,2),np.uint8)
    image = cv2.dilate(image, kernel, iterations=1)
    image = cv2.bitwise_not(image)
    return (image)
```

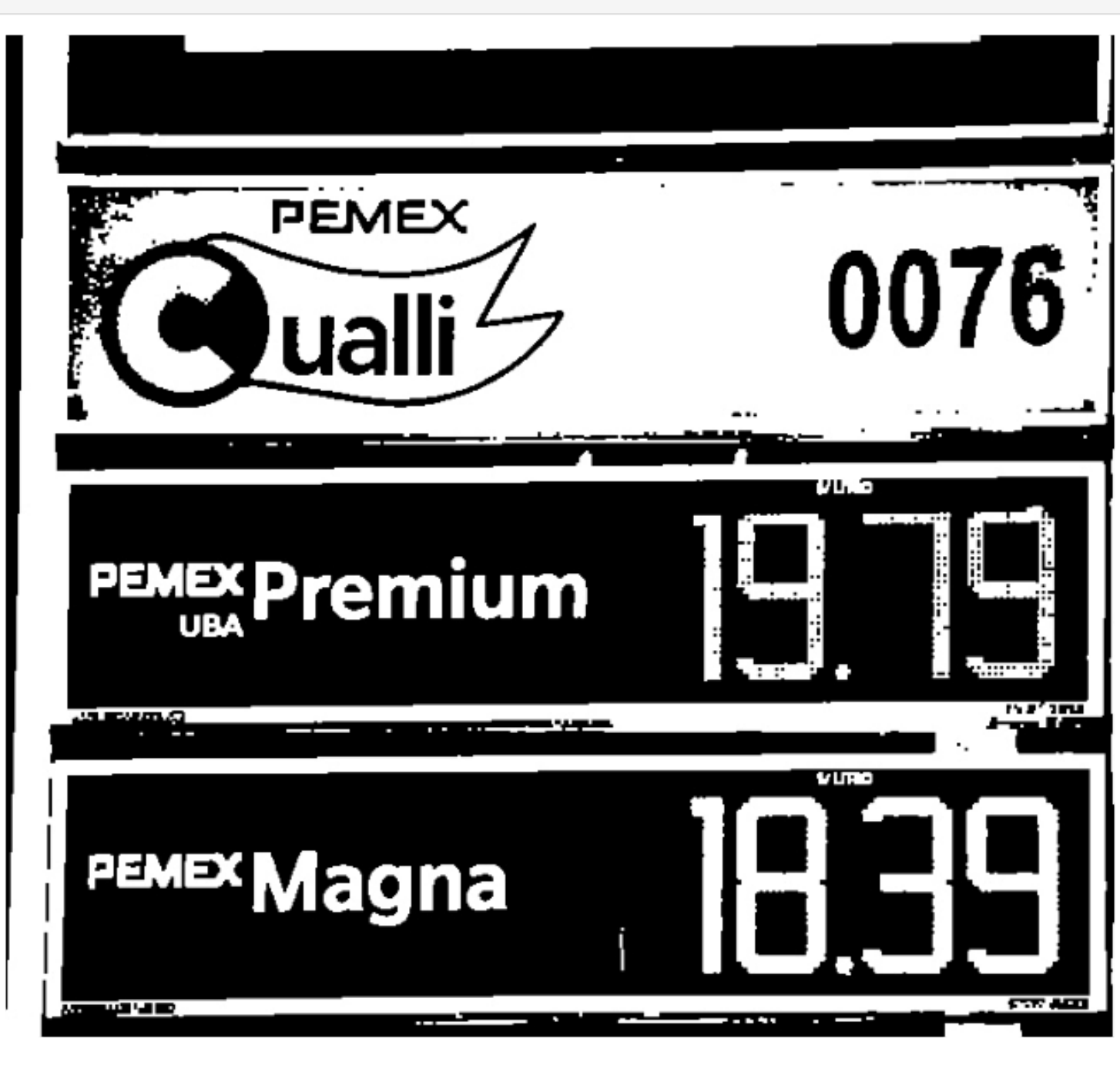
In [13]:
```
dilated_image = thick_font(eroded_image)
cv2.imwrite("/home/jos/Desktop/scriptsJos/ImagenesGaspre/dilated_image.jpg", dilated_image)
display("/home/jos/Desktop/scriptsJos/ImagenesGaspre/dilated_image.jpg")
```



Ahora realizamos un paso de quitar bordes par a hacer más fácil la lectura del texto y para que sea más fácil identificar los límites de cada uno de los textos individuales que se encuentran en la imagen.

In [14]:
```
def remove_borders(image):
    contours, heiarchy = cv2.findContours(image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cntsSorted = sorted(contours, key=lambda x:cv2.contourArea(x))
    cnt = cntsSorted[-1]
    x, y, w, h = cv2.boundingRect(cnt)
    crop = image[y:y+h, x:x+w]
    return (crop)
```

In [15]:
```
no_borders = remove_borders(dilated_image)
cv2.imwrite("/home/jos/Desktop/scriptsJos/ImagenesGaspre/no_borders.jpg", no_borders)
display('/home/jos/Desktop/scriptsJos/ImagenesGaspre/no_borders.jpg')
```



In [16]:
```
import cv2
import numpy as np
import easyocr
import matplotlib.pyplot as plt

def recognize_text(img_path):
    '''Cargamos la imagen, especificando el idioma y leemos el  texto.'''

    reader = easyocr.Reader(['en'])
    return reader.readtext(img_path)
```

Ahora realizamos la parte de reconocimiento de texto de la imagen y mostramos los resultados. Para cada texto encontrado se muestran las coordenadas de cada una de las esquinas del rectángulo que lo engloba, el texto y finalmente un nivel de confianza que va de 0 a 1 que nos indica que tan bien funciono el proceso para cada cadena de texto.

In [17]:
```
result = recognize_text('/home/jos/Desktop/scriptsJos/ImagenesGaspre/eroded_image.jpg')
```

```
CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
/home/jos/anaconda3/lib/python3.8/site-packages/torch/nn/functional.py:718: UserWarning: Named tensors and all their associated APIs are an experimental feature and subject to change. Plea
se do not use them for anything important until they are released as stable. (Triggered internally at  /pytorch/c10/core/TensorImpl.h:1156.)
  return torch.max_pool2d(input, kernel_size, stride, padding, dilation, ceil_mode)
```

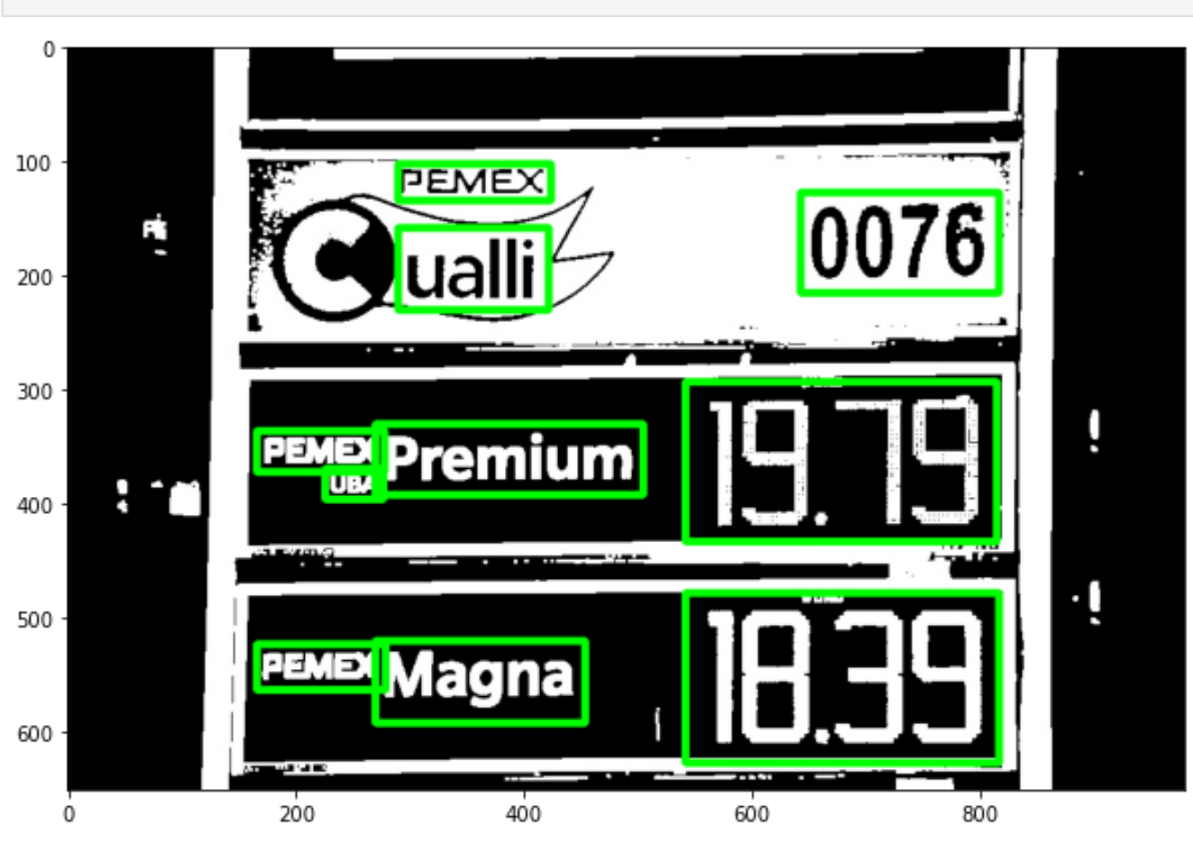
In [18]:
```
result
```

Out[18]:
```
[([[290, 102], [424, 102], [424, 134], [290, 134]],
  'PEMEX',
  0.991305135258939),
 ([[290, 158], [422, 158], [422, 230], [290, 230]],
  'ualli',
  0.9994742205906839),
 ([[643, 127], [817, 127], [817, 215], [643, 215]],
  '0076',
  0.9991176724433899),
 ([[166, 336], [278, 336], [278, 372], [166, 372]],
  'PEMeX',
  0.15463794239288153),
 ([[226, 369], [277, 369], [277, 396], [226, 396]],
  'UBVA',
  0.23394222557544708),
 ([[270, 330], [505, 330], [505, 392], [270, 392]],
  'Premium',
  0.9996717883211695),
 ([[542, 293], [815, 293], [815, 433], [542, 433]],
  '19.79',
  0.5362989081538122),
 ([[166, 523], [279, 523], [279, 563], [166, 563]],
  'PEMEX',
  0.8024230077439811),
 ([[270, 520], [454, 520], [454, 592], [270, 592]],
  'Magna',
  0.9999738770723775),
 ([[542, 478], [817, 478], [817, 627], [542, 627]],
  '1839',
  0.6383827924728394)]
```

Identificamos regiones de texto.

In [19]:
```
img = cv2.imread('/home/jos/Desktop/scriptsJos/ImagenesGaspre/eroded_image.jpg')
for detection in result:
  top_left = tuple([int(val) for val in detection[0][0]])
  bottom_right = tuple([int(val) for val in detection[0][2]])
  text = detection[1]
  font = cv2.FONT_HERSHEY_SIMPLEX
  img = cv2.rectangle(img,top_left,bottom_right, (0,255,0), 5)


plt.figure(figsize=(10,10))
plt.imshow(img)
plt.show()
```



Recuperamos el texto.

In [20]:
```
text=[None for _ in range(len(result))]
i=0
for detection in result:

    text[i] = detection[1]
    i=i+1
text
```

Out[20]:
```
['PEMEX',
 'ualli',
 '0076',
 'PEMeX',
 'UBVA',
 'Premium',
 '19.79',
 'PEMEX',
 'Magna',
 '1839']
```

Recuperamos los niveles de confianza y buscamos los resultados que tuvieron una confianza razonable para encontrar lo que buscamos, ya que los textos más pequeños que se encuentran en la imagen tendrán un nivel de confianza más bajo y no nos interesan.

In [21]:
```
confianza=[None for _ in range(len(result))]
i=0
for detection in result:

    confianza[i] = detection[2]
    i=i+1

confianza
```

Out[21]:
```
[0.991305135258939,
 0.9994742205906839,
 0.9991176724433899,
 0.15463794239288153,
 0.23394222557544708,
 0.9996717883211695,
 0.5362989081538122,
 0.8024230077439811,
 0.9999738770723775,
 0.6383827924728394]
```

In [22]:
```
indices=[index for index in range(len(confianza)) if confianza[index]>.5]
```

In [23]:
```
[text[i] for i in indices]
```

Out[23]:
```
['PEMEX', 'ualli', '0076', 'Premium', '19.79', 'PEMEX', 'Magna', '1839']
```