UNIVERSIDAD DE ORIENTE

NÚCLEO DE ANZOÁTEGUI

ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS

DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

PROYECTOS DIGITALES AVANZADOS

PRÁCTICA N°14:

IMAGEN EN DISPLAY SSD1306

Profesor:
Pedro Rene
Cabrera

Bachiller:
Claudia Rodríguez
C.I: 27.943.668

Barcelona, febrero de 2025

1. OBJETIVOS DE LA PRÁCTICA

1.1. Objetivo Principal: Mostrar una imagen (inicialmente en formato PNG, JPG o JPEG) en una pantalla OLED SSD1306, que es monocromática y tiene una resolución de 128x64 píxeles. La imagen debe ser convertida a formato BMP monocromático, almacenada en una tarjeta microSD y mostrada en la pantalla. Además, se debe implementar una funcionalidad de zoom para mostrar una región ampliada de la imagen con mayor nitidez.

1.2. Objetivos Específicos:

1.2.1. Convertir una imagen desde formatos comunes (PNG, JPG o JPEG) a un formato BMP monocromático de 1 bit por píxel, compatible con pantallas OLED SSD1306, teniendo en cuenta las limitaciones de resolución (128x64) y color de la pantalla (dos colores).

1.2.2. Almacenar la imagen convertida en formato BMP en una tarjeta microSD, asegurando que el archivo esté listo para ser leído por el Raspberry Pi Pico W.

1.2.3. Leer la imagen BMP almacenada en la tarjeta microSD y procesar sus datos para que puedan ser mostrados en la pantalla OLED.

1.2.4. Mostrar la imagen leída en la pantalla OLED SSD1306, ajustándola a la resolución de 128x64 píxeles sin distorsionarla.

1.2.5. Implementar la funcionalidad de zoom, permitiendo al usuario seleccionar una región de la imagen especificando coordenadas (X, Y) y las dimensiones de la región a mostrar, asegurando que la imagen ampliada sea nítida y detallada.

2. DESARROLLO

2.1. Planteamiento del Problema

Esta práctica N°14 se enfrenta a varios problemas que deben resolverse para lograr la visualización adecuada de una imagen en una pantalla OLED SSD1306. El primer reto es la conversión de la imagen a formato BMP monocromático, ya que las imágenes en formatos comunes como PNG, JPG o JPEG no son directamente compatibles con la pantalla.

Este formato BMP monocromático de 1 bit por píxel debe ser creado sin perder la calidad de la imagen, adaptándose a las limitaciones de resolución de la pantalla (128x64 píxeles) y su capacidad para representar solo dos colores que nos ofrece la pantalla OLED.

El siguiente problema radica en el almacenamiento de la imagen convertida en una tarjeta microSD. Una vez que la imagen ha sido transformada a BMP, debe ser almacenada correctamente en la tarjeta microSD, asegurando que el archivo sea accesible y pueda ser leído sin errores.

Posteriormente, el proceso de lectura de la imagen desde la tarjeta microSD representa otro desafío, pues se debe manejar la extracción de los datos del archivo BMP de forma precisa, especialmente teniendo en cuenta que las imágenes pueden ser grandes y el microcontrolador debe procesarlas correctamente para visualizarlas en la pantalla.

Luego, la imagen debe ser mostrada en la pantalla OLED SSD1306, que tiene una resolución limitada. El problema consiste en ajustar la imagen a las dimensiones de la pantalla sin distorsionarla y garantizar que se muestre centrada para una visualización óptima.

Finalmente, se debe de incorporar una función de zoom y mejora de nitidez, ya que la resolución de la pantalla y el tamaño de la imagen exigen una forma de ampliar una región seleccionada de la foto sin perder detalle. El reto aquí es cómo mantener la nitidez y mejorar el contraste de la imagen ampliada, ya que la ampliación de la imagen puede generar pixelado o pérdida de claridad.

2.2. Solución del Problema

La solución se divide en dos partes, la conversión de la imagen a formato BMP y mostrarla en la pantalla OLED con el zoom:

2.2.1. Solución para la conversión a BMP y zoom de la foto (script → escalado pc.py)

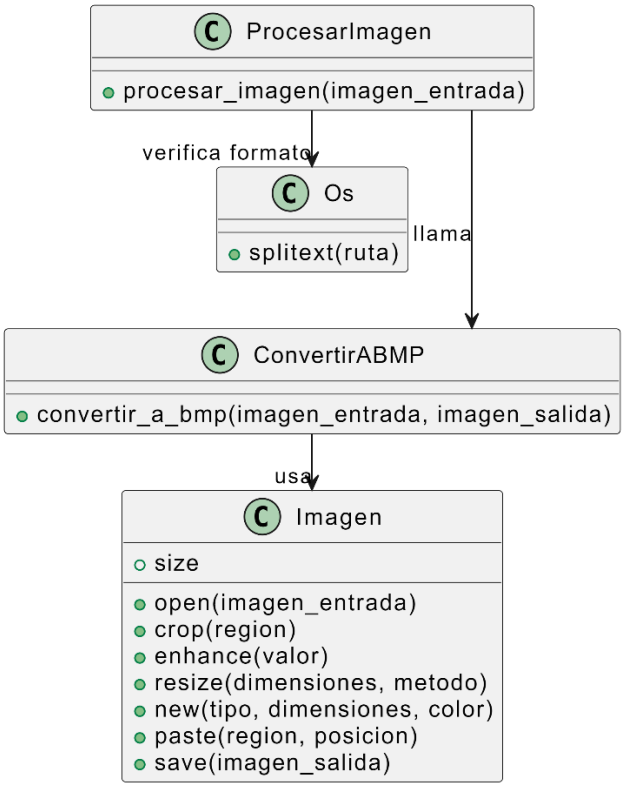
- Importación de bibliotecas: Se utiliza la biblioteca **PIL** para trabajar con imágenes. Es importante mencionar que la función ImageEnhance se utilizó para mejorar el contraste de la imagen y hacerla más nítida. La otra biblioteca utilizada es **os** pero ya ha sido usada en prácticas anteriores.
- Conversión de formato: Primero, el código abre la imagen original y obtiene sus dimensiones (ancho y alto). Luego, solicita al usuario que ingrese las coordenadas y el tamaño de la región de la imagen que desea ampliar. Una vez que se obtiene la región deseada, la imagen es recortada, y su contraste se aumenta para hacerla más nítida.
- Redimensionamiento y almacenamiento: La imagen es redimensionada a un tamaño adecuado (128x64 píxeles), se coloca en un lienzo blanco y se guarda como un archivo BMP. Este formato es compatible con pantallas monocromáticas como la SSD1306.

2.2.2. Solución para mostrar la foto en el SSD1306 (script → figura.py)

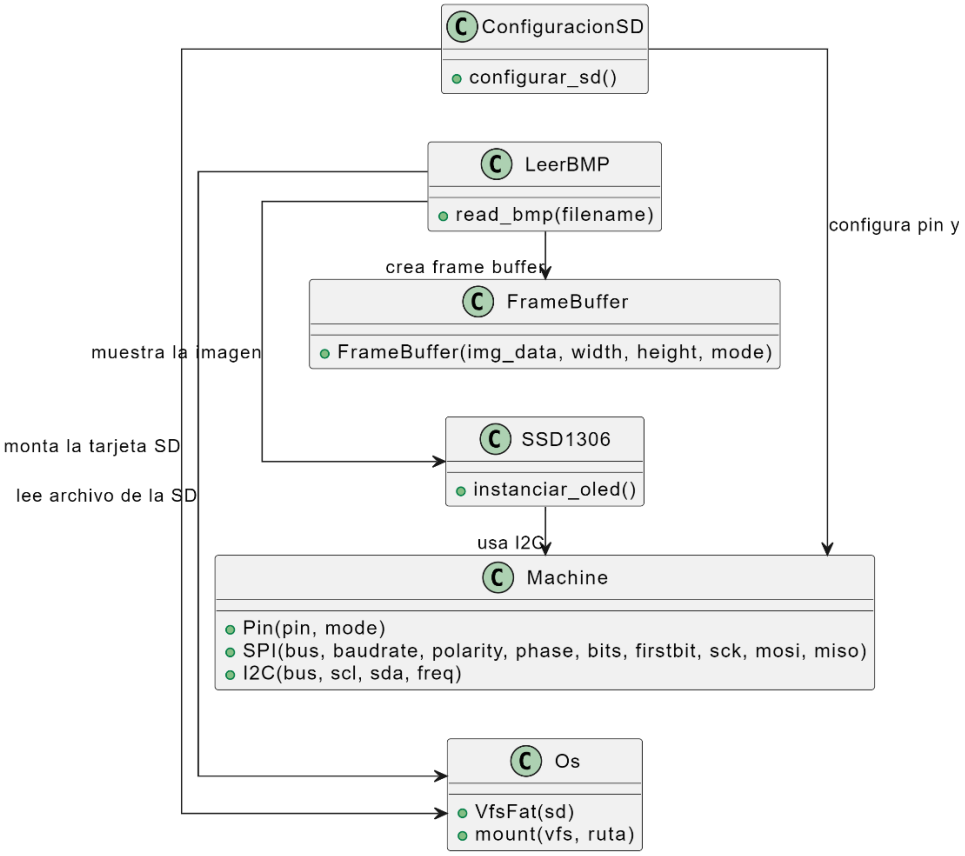
- Instanciamiento de la tarjeta SD y pantalla OLED: Ya esto se ha realizado en prácticas anteriores, no amerita explicación.
- Lectura de la imagen BMP: El código lee el archivo BMP desde la tarjeta SD, obtiene las dimensiones de la imagen y extrae los datos de la imagen (en formato binario). Luego, crea un FrameBuffer con estos datos, lo cual es necesario para mostrar imágenes en la pantalla OLED.
- Visualización de la imagen: La imagen cargada se dibuja en la pantalla OLED utilizando el método blit() y luego se muestra con show().

2.3. Diagrama UML

2.3.1. Código para convertir JPEG/JPG/PNG a BMP

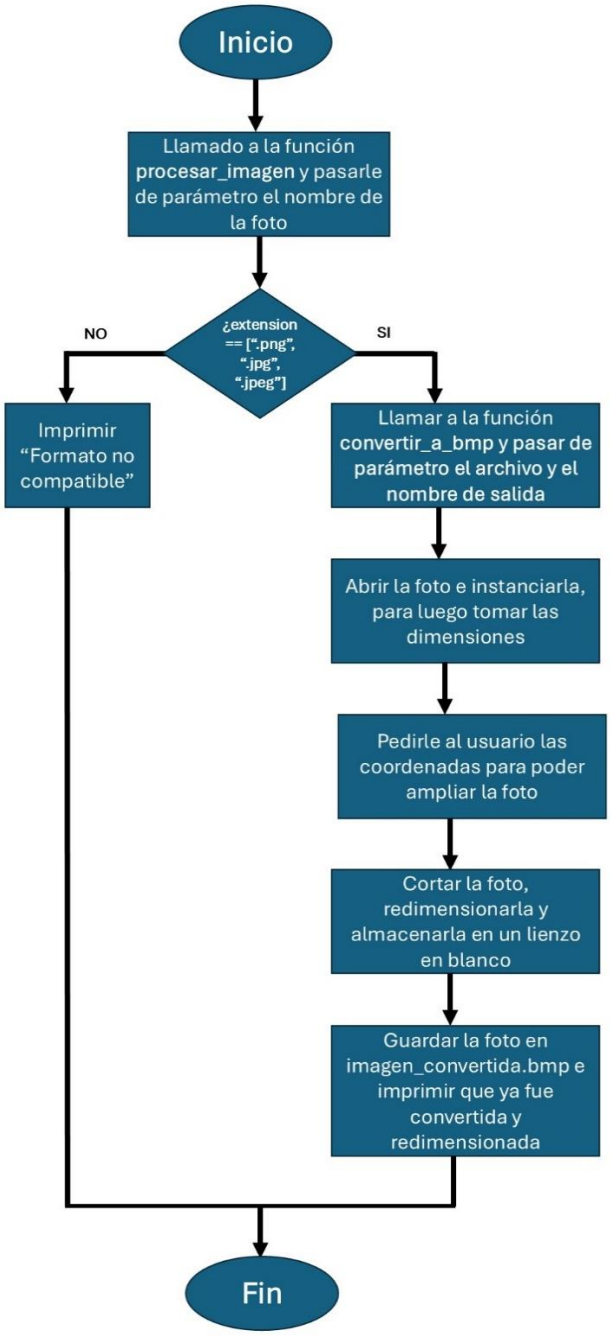


2.3.2. Código para convertir JPEG/JPG/PNG a BMP

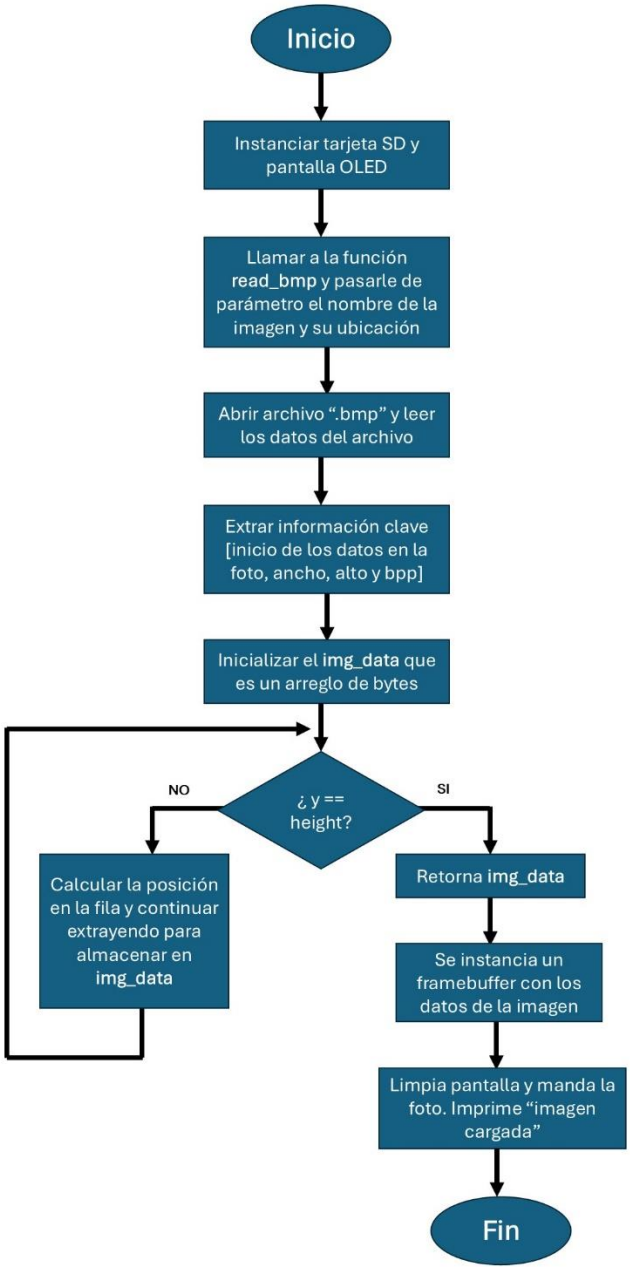


2.4. Diagrama de Flujo

2.4.1. Conversión y redimensión de foto



2.4.2. Mostrar en la pantalla OLED



3. ANEXOS

3.1. Código

3.1.1. Código para convertir de PNG/JPEG/JPG a BMP.

```
from PIL import Image, ImageOps,
ImageEnhance
import os

# -- convertir_a_bmp: funcion que convierte la
imagen a BMP -- #
def convertir_a_bmp(imagen_entrada,
imagen_salida):

    # abre la imagen y la instancia
    img = Image.open(imagen_entrada)
    # toma ancho y alto de la foto
    ancho, alto = img.size
    print(f"Tamaño original de la imagen:
{ancho}x{alto}")
    # pedir datos para hacer zoom
    print("Por favor, ingresa las coordenadas para
la region que deseas ampliar.")
    print(f"Las dimensiones de la imagen son
{ancho}x{alto}.")

    '''OJO POR ACA'''
    zoom_x = int(input("Coordenada X inicial (0 a
{0}): ".format(ancho - 1))) # coordenada X inicial
--> va desde 0 hasta el ancho-1
    zoom_y = int(input("Coordenada Y inicial (0 a
{0}): ".format(alto - 1)))  # coordenada Y inicial --
> va desde 0 hasta el alto-1
    zoom_w = int(input("Ancho de la region (1 a
{0}): ".format(ancho - zoom_x))) # ancho
p/ampliar --> va desde 1-ancho_restante
    zoom_h = int(input("Alto de la region (1 a {0}):
".format(alto - zoom_y)))   # alto p/ampliar  -->
va desde 1-alto_restante

    region = img.crop((zoom_x, zoom_y, zoom_x +
zoom_w, zoom_y + zoom_h)) # corta la region
con los parametros anteriores

    # aumentar contraste para mayor nitidez,
osea, que haya mas pixeles negros
    region =
ImageEnhance.Contrast(region).enhance(2.0)

    # redimensiona
    ratio = min(128 / zoom_w, 64 / zoom_h)
    nuevo_ancho = int(zoom_w * ratio)
    nuevo_alto = int(zoom_h * ratio)

    region = region.resize((nuevo_ancho,
nuevo_alto), Image.Resampling.LANCZOS)

    # esta es la imagen que se va a guardar, y lo
que queda vacio se rellena con 1 --> blanco
    lienzo = Image.new("1", (128, 64), 1)

    # calcular la posicion para centrar la imagen
redimensionada (opcional, pero se ve mejor
centrada)
    pos_x = (128 - nuevo_ancho) // 2
    pos_y = (64 - nuevo_alto) // 2

    # pega la imagen redimensionada en el lienzo
en blanco, centrada
    lienzo.paste(region.convert("1"), (pos_x,
pos_y))
    # guarda la imagen final en el formato
deseado
    lienzo.save(imagen_salida)
    print(f"Imagen {imagen_entrada} convertida,
redimensionada y guardada como
{imagen_salida}")

# -- procesar_imagen: verifica la extension de
la foto (png/jpeg/jpg) -- #
def procesar_imagen(imagen_entrada):
    extension =
os.path.splitext(imagen_entrada)[1].lower()
    if extension in [".png", ".jpg", ".jpeg"]:
        imagen_salida = "imagen_convertida.bmp"
        convertir_a_bmp(imagen_entrada,
imagen_salida)
    else:
        print("El formato de la imagen no es
compatible. Usa PNG o JPEG/JPG.")
procesar_imagen("2.JPG")
```

3.1.2. Código para mostrar la foto en el SSD1306

```
import framebuf
import ssd1306
import os
import sdcard
import machine

# config modulo sd
cs = machine.Pin(1, machine.Pin.OUT)
spi = machine.SPI(0, baudrate=1000000,
polarity=0, phase=0, bits=8,
firstbit=machine.SPI.MSB,
sck=machine.Pin(2), mosi=machine.Pin(3),
miso=machine.Pin(4))
sd = sdcard.SDCard(spi, cs)
vfs = os.VfsFat(sd)
os.mount(vfs, "/sd")

# instanciar ssd1306
i2c = machine.I2C(0, scl=machine.Pin(17),
sda=machine.Pin(16), freq=400000)
oled = ssd1306.SSD1306_I2C(128, 64, i2c)

# leer la foto desde la sd
def read_bmp(filename):
  # abre el archivo bmp en modo lectura
binaria
  with open(filename, "rb") as f:
    bmp = f.read()  # lee todo el contenido
del archivo

  # obtiene la posicion en la que comienzan
los datos de la imagen
  data_offset = int.from_bytes(bmp[10:14],
"little")  # posicion donde empiezan los datos

  # obtiene el ancho de la imagen a partir del
archivo bmp
  width = int.from_bytes(bmp[18:22], "little")
# ancho de la imagen

  # obtiene el alto de la imagen a partir del
archivo bmp
  height = int.from_bytes(bmp[22:26],
"little")    # alto de la imagen

  # obtiene los bits por pixel (bpp) a partir del
archivo bmp
  bpp = int.from_bytes(bmp[28:30], "little")
# bits por pixel

  # inicializa un arreglo para almacenar los
datos de la imagen
  img_data = bytearray()

  # calcula el tamaño de cada fila en bytes
(cada fila es un conjunto de pixeles)
  row_size = (width + 7) // 8  # bytes x fila

  for y in range(height):
    # calcula la posicion inicial de la fila (el
bmp almacena las filas de abajo hacia
arriba)
    row_start = data_offset + (height - 1 - y) *
row_size  # bmp almacena de abajo hacia
arriba

    # extrae la fila de datos y la agrega a
img_data

img_data.extend(bmp[row_start:row_start +
row_size])
  return img_data

img_data =
read_bmp("/sd/imagen_convertida.bmp")

# crea un frame buffer con los datos de la
imagen
fb = framebuf.FrameBuffer(img_data, 128,
64, framebuf.MONO_HLSB)
oled.fill(0)
oled.blit(fb, 0, 0)
oled.show()
print("imagen cargada desde bmp y
mostrada en la oled.")
```

3.2. Bibliografía

INTRODUCCIÓN A PILLOW (PYTHON IMAGING LIBRARY) (PARTE 1). Enlace: https://www.youtube.com/watch?v=YL8_aAh0v5k

SSD1306 pantalla OLED con Arduino y ESP8266 I2C. Enlace: https://programarfacil.com/blog/arduino-blog/ssd1306-pantalla-oled-con-arduino/

framebuf — frame buffer manipulation. Enlace: https://docs.micropython.org/en/latest/library/framebuf.html

Image file formats. Enlace: https://pillow.readthedocs.io/en/stable/handbook/image-file-formats.html

Mejora de Nitidez de una imagen a escala de grises con Python 3. Enlace: https://pythoneyes.wordpress.com/2017/08/16/mejora-de-nitidez-de-una-imagen-a-escala-de-grises-con-python-3/

BMP file format. Enlace: https://gibberlings3.github.io/iesdp/file_formats/ie_formats/bmp.htm

3.3. Diagrama circuital