```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from scipy.signal import butter, lfilter

def image_to_binary(image_path):
    img = Image.open(image_path).convert('L')  # Convert to grayscale
    img = img.resize((64, 64))  # Resize for transmission
    img_array = np.array(img, dtype=np.uint8)
    binary_data = np.unpackbits(img_array)  # Convert pixel values to binary
    return binary_data, img_array.shape

def bpsk_modulation(bits):
    return 2 * bits - 1  # Convert 0 -> -1, 1 -> +1

def add_noise(signal, noise_level=0.3):
    noise = np.random.normal(0, noise_level, signal.shape)
    return signal + noise

def bpsk_demodulation(received_signal):
    return (received_signal > 0).astype(np.uint8)  # Convert back to binary

def binary_to_image(binary_data, img_shape):
    pixel_values = np.packbits(binary_data)  # Convert binary back to pixels
    img_array = pixel_values.reshape(img_shape)  # Reshape to original image shape
    return Image.fromarray(img_array, mode='L')

def transmit_image(image_path):
    binary_data, img_shape = image_to_binary(image_path)
    modulated_signal = bpsk_modulation(binary_data)
    transmitted_signal = add_noise(modulated_signal, noise_level=0.3)
    received_signal = bpsk_demodulation(transmitted_signal)
    reconstructed_image = binary_to_image(received_signal, img_shape)

    return reconstructed_image

def display_images(original_path, reconstructed_image):
    plt.figure(figsize=(10, 4))
    plt.subplot(1, 2, 1)
    plt.title("Original Image")
    plt.imshow(Image.open(original_path).convert('L').resize((64, 64)), cmap='gray')
    plt.subplot(1, 2, 2)
    plt.title("Received Image After Transmission")
    plt.imshow(reconstructed_image, cmap='gray')
    plt.show()




//this stuff in main

from cubesat_communication import transmit_image, display_images
from picamera2 import Picamera2
import time

def capture_image(filename="captured_image.jpg"):
    picam = Picamera2()
    still_config = picam.create_still_configuration(main={"size": (1280, 720), "format": "RGB888"})
    picam.configure(still_config)
    picam.start()
    time.sleep(2)  # Allow camera to adjust exposure
    picam.capture_file(filename)
    picam.stop()
    return filename

image_path = capture_image()
received_image = transmit_image(image_path)
display_images(image_path, received_image)
```