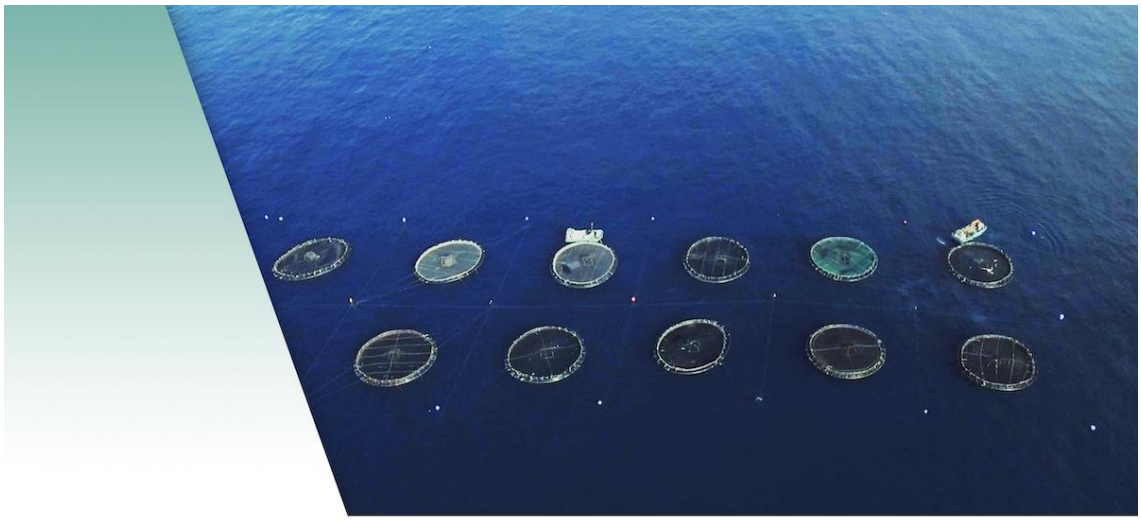
Proyecto SIRENA:

Sistema Remoto Integral de Monitorización, Detección y Predicción de Riesgos de Eventos Marinos Potencialmente Nocivos de Origen Natural o Antrópico en Áreas Off-Shore Destinadas a la Acuicultura



Informe de Resultados / Fuente de Verificación

R1.2.2 Script de procesamiento automático de imágenes de satélite Sentinel 2

Fecha: 23/05/2025

Entidad Responsable: Universidad de Las Palmas de Gran Canaria (ULPGC)

Este proyecto se desarrolla con la colaboración de la Fundación Biodiversidad del Ministerio para la Transición Ecológica y el Reto Demográfico, a través del Programa Pleamar, y se cofinancia por la Unión Europea por el FEMPA (Fondo Europeo Marítimo, de Pesca y de Acuicultura).

Indíce

Las opiniones y documentación aportadas en esta publicación son de exclusiva responsabilidad de la persona o personas que ostenten su autoría y no reflejan necesariamente los puntos de vista de las entidades que apoyan económicamente el proyecto.

1. Introducción

En el marco del **Resultado R1.2.2** del proyecto **SIRENA**, el **Instituto Universitario ECOAQUA de la Universidad de Las Palmas de Gran Canaria (ULPGC)** ha desarrollado un **script en Python para el procesamiento automático de imágenes satelitales Sentinel-2**. Este script tiene como objetivo optimizar el flujo de trabajo técnico mediante la automatización de las principales etapas de preprocesamiento y análisis de datos ópticos.

El código permite realizar tareas clave como la **corrección atmosférica**, **recorte geográfico**, y la **generación de índices espectrales** (como NDVI o MCI), facilitando así la extracción de información ambiental relevante a partir de imágenes multiespectrales. Este recurso técnico contribuye directamente a los objetivos del proyecto al permitir un análisis más ágil y reproducible de las condiciones del entorno marino-costero.

Este desarrollo constituye una **fuente de verificación técnica** del proyecto, demostrando la capacidad del equipo para crear herramientas innovadoras en el ámbito de la teledetección aplicada a la sostenibilidad marina.

.

2. Script de procesamiento Sentinel 2

A continuación, se presenta el código fuente correspondiente a este resultado.

```
# ================================================
# SCRIPT DE PROCESAMIENTO CREADO PARA EL PROYECTO SIRENA
# POR INSTITUTO UNIVERSITARIO ECOAQUA - ULPGC
# ================================================


import os
from snappy import ProductIO, GPF, HashMap, jpy
from glob import glob

# Coordenadas del subset
lonmin = -15.557
latmin = 27.429
lonmax = -15.657
latmax = 27.629

# Directorios
input_folder = r'C:\Sirena\Descargas'
output_folder = os.path.join(input_folder, 'sent2ndvi')
os.makedirs(output_folder, exist_ok=True)

# Cargar operadores de SNAP
GPF.getDefaultInstance().getOperatorSpiRegistry().loadOperatorSpis()

# Buscar archivos Sentinel-2 .SAFE
safe_files = glob(os.path.join(input_folder, 'S2*.SAFE'))

for safe_path in safe_files:
    print(f'Procesando: {safe_path}')
```

```
    # Leer producto
    product = ProductIO.readProduct(safe_path)

    # Paso 1: Subset geográfico
    print(' -> Subset...')
    subset_params = HashMap()
    subset_params.put('geoRegion', f"POLYGON(({lonmin} {latmin}, {lonmax}
{latmin}, {lonmax} {latmax}, {lonmin} {latmax}, {lonmin} {latmin}))")
    subset_params.put('outputImageScaleInDb', False)
    subset = GPF.createProduct('Subset', subset_params, product)

    # Paso 2: Máscara de tierra
    print(' -> Land/Sea Mask...')
    land_params = HashMap()
    land_params.put('landMask', True)
    land_params.put('useSRTM', True)
    land_params.put('geometry', None)
    landmasked = GPF.createProduct('Land-Sea-Mask', land_params, subset)

    # Paso 3: Filtrado de nubes usando la clasificación de escena
    print(' -> Filtrar nubes (SCL)...')
    band_math_params = HashMap()
    band_math_params.put('name', 'SCL_CLOUDMASK')
    band_math_params.put('expression', 'SCL != 3 && SCL != 8 && SCL != 9 &&
SCL != 10')  # Excluir sombra, nubes medianas y altas
    band_math_params.put('noDataValue', 0)
    cloud_filtered = GPF.createProduct('BandMaths', band_math_params,
landmasked)

    # Paso 4: Cálculo de NDVI
    print(' -> Calculando NDVI...')
    ndvi_params = HashMap()
    ndvi_params.put('name', 'NDVI')
```

```
    ndvi_params.put('expression', '(B8 - B4) / (B8 + B4)')  # B8 = NIR, B4 =
Red
    ndvi = GPF.createProduct('BandMaths', ndvi_params, cloud_filtered)

    # Paso 5: Filtrar NDVI > 0.5
    print(' -> Filtrando NDVI > 0.5...')
    threshold_params = HashMap()
    threshold_params.put('name', 'NDVI_gt_05')
    threshold_params.put('expression', 'NDVI > 0.5')
    thresholded = GPF.createProduct('BandMaths', threshold_params, ndvi)

    # Paso 6: Guardar resultado
    basename = os.path.basename(safe_path).replace('.SAFE', '_ndvi.dim')
    output_path = os.path.join(output_folder, basename)
    print(f' -> Guardando: {output_path}')
    ProductIO.writeProduct(thresholded, output_path, 'BEAM-DIMAP')

print("Procesamiento Sentinel-2 finalizado.")
```