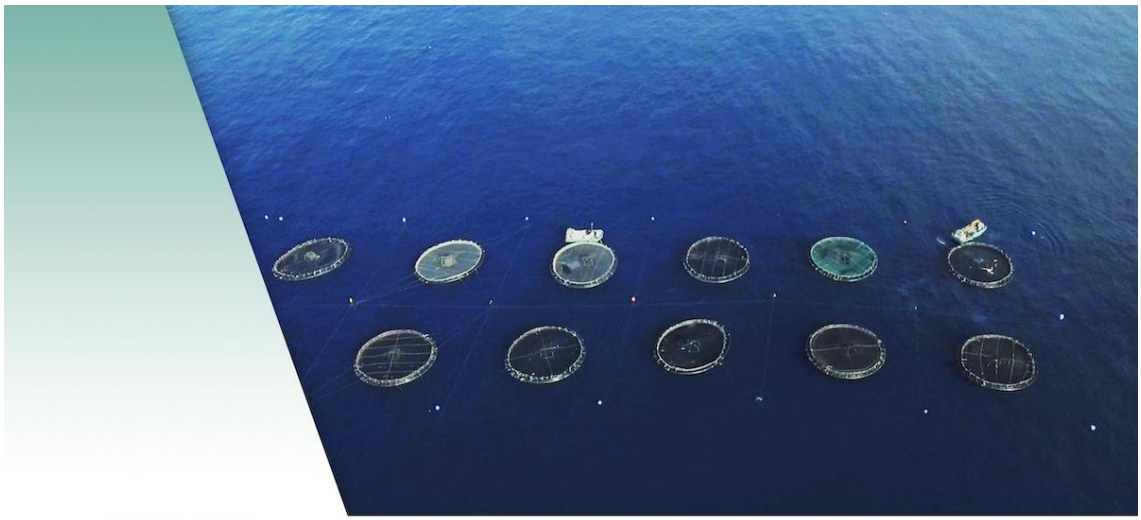
Proyecto SIRENA:

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

R1.2.1 Script de procesamiento automático de imágenes de satélite Sentinel 1

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

Como parte del **Resultado R1.2.1** 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-1**. Este desarrollo tiene como finalidad facilitar la transformación de datos brutos en productos geoespaciales útiles para el análisis de variables marinas, permitiendo un seguimiento más eficiente y detallado del entorno costero.

El script permite la automatización de tareas de preprocesamiento como la calibración radiométrica, corrección geométrica y generación de productos derivados a partir de imágenes radar, mejorando significativamente la eficiencia y reproducibilidad del flujo de trabajo técnico del proyecto.

Este código constituye una **fuente de verificación** fundamental del avance técnico del proyecto y demuestra la capacidad de generar herramientas innovadoras para la gestión y monitorización ambiental.

.

2. Script de procesamiento Sentinel 1

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
from glob import glob

# Parámetros de área para el subset
lonmin = -15.557
latmin = 27.429
lonmax = -15.657
latmax = 27.629

# Ruta de trabajo
input_folder = r'C:\Sirena\Descargas'
output_folder = os.path.join(input_folder, 'oilspilldetection')
os.makedirs(output_folder, exist_ok=True)

# Registrar operadores
GPF.getDefaultInstance().getOperatorSpiRegistry().loadOperatorSpis()

# Lista de archivos .SAFE
safe_files = glob(os.path.join(input_folder, '*.SAFE'))

for safe_path in safe_files:
```

```
    print(f'Procesando: {safe_path}')

    # Cargar producto original
    product = ProductIO.readProduct(safe_path)

    # Paso 1: Calibración
    print(' -> Calibrando...')
    parameters = HashMap()
    parameters.put('outputSigmaBand', True)
    parameters.put('sourceBands', 'Intensity_VV')  # Usa también
'Intensity_VH' si lo deseas
    parameters.put('selectedPolarisations', 'VV')
    parameters.put('calibrationModel', 'Sigma0')
    calibrated = GPF.createProduct('Calibration', parameters, product)

    # Paso 2: Subset
    print(' -> Aplicando 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, calibrated)

    # Paso 3: Speckle Filter
    print(' -> Aplicando filtro speckle...')
    speckle_params = HashMap()
    speckle_params.put('filter', 'Lee')
    speckle = GPF.createProduct('Speckle-Filter', speckle_params, subset)

    # Paso 4: Land/Sea Mask
    print(' -> Aplicando máscara de tierra...')
    landmask_params = HashMap()
    landmask_params.put('landMask', True)
```

```
    landmask_params.put('useSRTM', True)
    landmask_params.put('geometry', None)
    landmasked = GPF.createProduct('Land-Sea-Mask', landmask_params, speckle)

    # Paso 5: Oil Spill Detection
    print(' -> Detectando derrames...')
    oilspill = GPF.createProduct('Oil-Spill-Detection', HashMap(), landmasked)

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

print('Procesamiento completo.')
```