



UNIVERSIDAD NACIONAL AUTÓNOMA  
DE MÉXICO

---

---

FACULTAD DE INGENIERÍA

**Procesamiento automatizado de productos meteorológicos  
derivados de satélites geostacionarios y de órbita polar.**

**T E S I S**

Que para obtener el título de:

**INGENIERO GEOMÁTICO**

PRESENTA:

**Uriel de Jesús Mendoza Castillo**

DIRECTORA DE TESIS:

**M.C. María Elena Osorio Tai**



Ciudad Universitaria, Cd. Mx., 2019



Universidad Nacional  
Autónoma de México

Dirección General de Bibliotecas de la UNAM

**Biblioteca Central**



**UNAM – Dirección General de Bibliotecas**  
**Tesis Digitales**  
**Restricciones de uso**

**DERECHOS RESERVADOS ©**  
**PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL**

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.



# Agradecimientos

A la gran comunidad de desarrolladores de software libre que comparte código y resuelve dudas a través de la red, permitiendo que el aprendizaje se desarrolle de manera colectiva.

A la Universidad Autónoma Chapingo (UACH) y a la Universidad Nacional Autónoma de México (UNAM), por brindarme una educación gratuita y de calidad, así como a los grandes profesores, compañeros y amigos que pude conocer durante estas etapas.

Al Ing. Erick de Valle Salgado, a la Ing. Ana Lilia Salas Alvarado, al M.I Juan Manuel Nuñez Hernández y a la Dra. Griselda Berenice Hernández Cruz de la Facultad de Ingeniería (FI), por haber aceptado ser jurado de este trabajo y haber dedicado parte de su tiempo en hacer revisiones y correcciones.

A la Dra. Olivia Salmerón García, a la Dra. Lilia de Lourdes Manzo Delgado y al M.C. Alejandro Aguilar Sierra del Instituto de Geografía (IGg), por sus grandes aportaciones a este trabajo en su etapa inicial y final, sus conocimientos altamente especializados fueron clave para el desarrollo de este.

Al Dr. Jorge Prado Molina y la Dra. Gabriela Gómez Rodríguez del IGg, por invitarme a ser parte del proyecto Laboratorio Nacional de la Tierra (LANOT), así como el permitirme hacer uso de los equipos del laboratorio para el desarrollo de este trabajo, agradezco también la confianza depositada en mí para la propuesta de soluciones.

A la M.C. María Elena Osorio Tai, por haber aceptado ser la directora de este trabajo y haber dedicado gran parte de su tiempo para realizar correcciones y aportaciones, una persona altamente dedicada la cual admiro desde inicios de la carrera, siendo uno de mis mayores ejemplos a seguir.

A mis amigos y familiares, a mi hermana Elisa por abrirme horizontes y enseñarme el mundo desde que era pequeño, a mi hermano Jacob por compartir con él los momentos más divertidos y felices de mi infancia.

Y sobre todo, por qué el trabajo más difícil del mundo, es también el mejor trabajo del mundo...

Gracias mamá.



# Índice general

<b>Agradecimientos</b>	<b>III</b>
<b>Abreviaturas</b>	<b>IX</b>
<b>Resumen</b>	<b>XI</b>
<b>Índice de figuras</b>	<b>XII</b>
<b>Índice de tablas</b>	<b>XVI</b>
<b>1. Introducción</b>	<b>1</b>
1.1. Antecedentes . . . . .	3
1.2. Planteamiento del problema . . . . .	8
1.3. Objetivos . . . . .	9
<b>2. Observación meteorológica satelital</b>	<b>11</b>
2.1. Satélites meteorológicos . . . . .	11
2.2. Satélites geoestacionarios . . . . .	24
2.2.1. GOES-16 . . . . .	24
2.3. Satélites de Órbita Polar . . . . .	28
2.3.1. Suomi-NPP . . . . .	28
2.3.2. Sentinel-2 . . . . .	31
<b>3. Productos meteorológicos satelitales</b>	<b>35</b>
3.1. Variables meteorológicas . . . . .	35
3.1.1. Land Surface Temperature . . . . .	36

3.1.2. Sea Surface Temperature . . . . .	40
3.2. Compuestos RGB . . . . .	46
3.2.1. True Color . . . . .	47
3.2.2. Fire Temperature . . . . .	50
<b>4. Metodología</b>	<b>53</b>
4.1. Extracción . . . . .	54
4.2. Tratamiento . . . . .	63
4.3. Georreferencia . . . . .	68
4.4. Reproyección . . . . .	75
4.5. Mapeo . . . . .	76
4.6. Procesos . . . . .	79
4.6.1. Proceso 1. Estaciones meteorológicas EMAS, GOES-16 y Sentinel-2 . . . . .	80
4.6.2. Proceso 2. Boyas meteorológicas NOAA, Suomi-NPP y Sentinel-2 . . . . .	86
4.6.3. Proceso 3. Puntos de calor, GOES-16, Suomi-NPP y Sentinel-2 . . . . .	90
<b>5. Resultados</b>	<b>95</b>
5.1. Proceso 1. Estaciones meteorológicas EMAS, GOES-16 y Sentinel-2 . . . . .	95
5.2. Proceso 2. Boyas meteorológicas NOAA, Suomi-NPP y Sentinel-2 . . . . .	99
5.3. Proceso 3. Puntos de calor, GOES-16, Suomi-NPP y Sentinel-2 . . . . .	102
<b>6. Conclusiones</b>	<b>107</b>
<b>Anexos</b>	<b>109</b>
<b>A. Formatos</b>	<b>111</b>
A.1. NetCDF4 . . . . .	111
A.2. GeoTIFF . . . . .	112
A.3. WMS . . . . .	112
<b>B. Scripts Bash</b>	<b>113</b>
<b>C. Scripts Python</b>	<b>115</b>
C.1. Funciones genéricas . . . . .	115

C.2. Datos de entrada . . . . .	119
C.3. Procesamiento de datos . . . . .	123
C.4. Scripts secundarios . . . . .	130
C.5. Scripts principales . . . . .	138
<b>D. Sentencias Cron</b>	<b>145</b>
<b>Bibliografía</b>	<b>147</b>





# Abreviaturas

<b>ABI</b> Advanced Baseline Imager	<b>GeoTIFF</b> Geographic Tagged Image File Format	<b>NIR</b> Near Infrared
<b>AHI</b> Advanced Himawari Imager	<b>GFS</b> Global Forecast System	<b>NOAA</b> National Oceanic and Atmospheric Administration
<b>AIT</b> Algorithm Integration Team	<b>GIF</b> Graphics Interchange Format	<b>NPP</b> National Polar-orbiting Partnership
<b>AVHRR</b> Advanced Very High Resolution Radiometer	<b>GIS</b> Geographic Information System	<b>OGR</b> OpenGIS Simple Features Reference
<b>BT</b> Brightness Temperature	<b>GMT</b> Greenwich Mean Time	<b>PNG</b> Portable Network Graphics
<b>CE</b> Comisión Europea	<b>GMT</b> Generic Mapping Tools	<b>POES</b> Polar Orbiting Environmental Satellites
<b>CGAL</b> Computational Geometry Algorithms Library	<b>GNU</b> GNU's Not Unix	<b>QC</b> Quality Control
<b>CMI</b> Cloud and Moisture Imagery	<b>GOES</b> Geostationary Operational Environmental Satellite	<b>RGB</b> Red Green Blue
<b>CONUS</b> Continental United States	<b>GRIB</b> Gridded Binary	<b>RTM</b> Radiative Transfer Modeling
<b>CRAN</b> Comprehensive R Archive Network	<b>HDF</b> Hierarchical Data Format	<b>SDE</b> Spatial Database Engine
<b>CRTM</b> Community Radiative Transfer Model	<b>HTML</b> HyperText Markup Language	<b>SDR</b> Sensor Data Records
<b>CSPP</b> Community Satellite Processing Package	<b>HTTP</b> Hypertext Transfer Protocol	<b>SIG</b> Sistema de Información Geográfica
<b>CSV</b> Comma Separated Values	<b>IDL</b> Interface Definition Language	<b>SMN</b> Servicio Meteorológico Nacional
<b>DOC</b> Department of Commerce of the United States	<b>IGBP</b> International Geosphere-Biosphere Programme	<b>SPOT</b> Satellite Pour l'Observation de la Terre
<b>DOD</b> Department of Defense	<b>JPG</b> Joint Photographic Experts Group	<b>SQL</b> Structured Query Language
<b>DQF</b> Data Quality Flag	<b>JPSS</b> Joint Polar Satellite System	<b>SST</b> Sea Surface Temperature
<b>EDR</b> Environmental Data Records	<b>JTS</b> Java Topology Suite	<b>SWIR</b> Short Wave Infrared
<b>EMA</b> Estaciones Meteorológicas Automáticas	<b>LANOT</b> Laboratorio Nacional de Observación de la Tierra	<b>TC</b> True Color
<b>EMC</b> Error Medio Cuadrático	<b>LIDAR</b> Light Detection and Ranging o Laser Imaging Detection and Ranging	<b>TLE</b> Two Line Element
<b>EPSCG</b> European Petroleum Survey Group	<b>LST</b> Land Surface Temperature	<b>URL</b> Uniform Resource Locator
<b>ESA</b> European Space Agency	<b>MODIS</b> Moderate Resolution Imaging Spectroradiometer	<b>VIIRS</b> Visible Infrared Imaging Radiometer Suite
<b>FIRMS</b> Fire Information for Resource Management System	<b>MSI</b> Multispectral Instrument	<b>WKT</b> Well Known Text
<b>FT</b> Fire Temperature	<b>NASA</b> National Aeronautics and Space Administration	<b>WMS</b> Web Map Service
<b>FTP</b> File Transfer Protocol	<b>NDVI</b> Normalized Difference Vegetation Index	<b>XLS</b> Extensible Stylesheet Language
<b>GDAL</b> Geospatial Data Abstraction Library	<b>NETCDF</b> Network Common Data Form	
<b>GEOS</b> Geometry Engine Open Source		



# Resumen

Se describe el desarrollo de una metodología para la integración y procesamiento automático de datos satelitales, provenientes de tres diferentes tipos de satélites meteorológicos y de observación terrestre, tanto de órbita polar (Suomi-NPP y Sentinel-2) como geostacionaria (GOES-16), mediante el uso de módulos especializados para el manejo de datos espaciales, disponibles en el lenguaje de programación Python. La metodología describe las etapas de extracción, tratamiento, georreferencia, reproyección y mapeo en cada formato de dato utilizado. Los datos fueron adquiridos en tres diferentes formatos, los cuales están asociados a diferentes niveles de adquisición, por ello en cada etapa de la metodología se utilizaron diferentes módulos. Los formatos ocupados fueron NetCDF4 para GOES-16, GeoTIFF para Suomi-NPP y WMS para Sentinel-2

Utilizando esta metodología, se desarrollaron tres procesos que integraron los productos de cada satélite. Los productos ocupados fueron LST y SST, que están asociados a variables meteorológicas, y los productos TC, FT y SWIR, que están asociados a composiciones RGB. En la metodología se crearon scripts con funciones genéricas que optimizaron su integración en cada proceso, lo cual permitió que la automatización de los procesos se desarrollara de manera simple.

El primer proceso genera una gráfica de comparación de los datos de LST de GOES-16 con los datos de temperatura del aire de las EMAS, durante las últimas 24 hrs, añadiendo el compuesto TC del último paso de Sentinel-2 en la ubicación de la EMA y los datos de LST. Este proceso mostró que los dos valores siguen la misma tendencia a lo largo de las 00:00 a las 14:00 GMT en la mayoría de las estaciones.

El segundo proceso genera una gráfica de comparación de los datos de SST de Suomi-NPP con los datos de temperatura superficial del mar de boyas meteorológicas NOAA, durante las últimas 24 hrs, añadiendo el compuesto TC del último paso de Sentinel-2 en la ubicación de la boya y los datos de SST. Este proceso mostro que el valor de SST, varía en promedio de 0.1 a 0.2 grados de diferencia con respecto al valor de la boya.

El tercer proceso genera las composiciones de TC de GOES-16, Suomi-NPP y Sentinel-2, FT de GOES-16 y Suomi-NPP y SWIR de Sentinel-2, en la ubicación de puntos de calor obtenidos de GOES-16. Este proceso ayudo a la eliminación de puntos asociados a incendios activos y a la verificación de estos.



# Índice de figuras

1.1. Imagen de temperatura de nubes con datos del sensor MODIS/Aqua. . . . .	4
1.2. Mapa de precipitación acumulada de 1410 estaciones del SMN. . . . .	5
2.1. Elementos espaciales básicos de los sensores pasivos. . . . .	12
2.2. Órbita Geoestacionaria y órbita Geosincrónica. . . . .	15
2.3. Trayectoria de la órbita polar. . . . .	17
2.4. Calendario oficial de próximos satélites del programa GOES. . . . .	25
2.5. Cobertura del segmento espacial del programa GOES. . . . .	25
2.6. Regiones de escaneo del satélite GOES-16. . . . .	28
2.7. Calendario oficial de próximos satélites del programa de satélites meteorológicos de órbita polar de la NOAA. . . . .	29
2.8. Regiones de escaneo del satélite Suomi-NPP . . . . .	29
2.9. Calendario oficial de próximos satélites Sentinel. . . . .	31
2.10. Regiones de escaneo del satélite Sentinel-2. . . . .	32
3.1. Compuesto del producto LST de MODIS/Terra global de 2003 a 2009. . . . .	36
3.2. Diagrama de flujo del procesamiento del producto LST de VIIRS/Suomi-NPP. . . . .	37
3.3. Diagrama de flujo del procesamiento del producto LST de ABI/GOES-16. . . . .	39
3.4. Compuesto del producto SST de MODIS/Aqua global del mes de Julio del 2002. . . . .	41

3.5. Diagrama de flujo del procesamiento del producto SST de VIIRS/Suomi-NPP. . . . .	42
3.6. Diagrama de flujo del procesamiento del producto SST de ABI/GOES-16. . . . .	44
3.7. Combinación de algoritmo SST de regresión e inversión para ABI/GOES-16. . . . .	45
3.8. Cubo de colores RGB. . . . .	46
3.9. RGB TC de VIIRS/Suomi-NPP. . . . .	47
3.10. Generación de verde sintético con bandas de ABI/GOES-16. . . . .	48
3.11. Producto RGB True color de ABI/GOES-16. . . . .	49
3.12. Producto RGB TC de MSI/Sentinel-2. . . . .	49
3.13. Producto RGB FT de ABI/GOES-16. . . . .	51
3.14. Producto RGB FT de VIIRS/Suomi-NPP. . . . .	51
3.15. Producto RGB SWIR de MSI/Sentinel-2. . . . .	52
4.1. Distribución del uso de módulos de Python usados en la metodología. . . . .	54
4.2. Estructura de datos en los diccionarios de Python. . . . .	57
4.3. Valores enmascarados de las esquinas del Full Disk. . . . .	59
4.4. Expresiones y forma de selección en arreglos <i>Numpy</i> . . . . .	64
4.5. Operaciones compatibles en arreglos <i>Numpy</i> . . . . .	65
4.6. Ubicación de valores en arreglos <i>Numpy</i> . . . . .	66
4.7. Mapa global generado con <i>CartoPy</i> a partir de los parámetros de proyección de los datos de GOES-16. . . . .	77
4.8. Mapa global generado con <i>CartoPy</i> a partir de los parámetros de proyección de los datos de Suomi-NPP. . . . .	78
4.9. Mapa global generado con <i>CartoPy</i> a partir de los parámetros de proyección de los datos de Sentinel-2. . . . .	78
4.10. Página del SMN con el mapa de EMAS. . . . .	81
4.11. Estaciones EMAS seleccionadas. . . . .	82
4.12. Patrón de distribución del PNG de integración final del proceso 1. . . . .	84

4.13. Diagrama de flujo del proceso 1. . . . .	85
4.14. Página de la National Data Buoy Center NOAA con el mapa de boyas meteorológicas. . . . .	87
4.15. Etiqueta de la página donde se extrajeron datos de la boya mediante Web Scrupting. . . . .	87
4.16. Patrón de distribución del PNG de integración final del proceso 2. . . . .	88
4.17. Diagrama de flujo del proceso 2. . . . .	89
4.18. Patrón de distribución del PNG de integración final del proceso 3. . . . .	92
4.19. Diagrama de flujo del proceso 3. . . . .	93
5.1. Resultado de proceso 1, con LST de ABI/GOES-16 procesado en la ubicación de la estación EMA. .	96
5.2. Resultado de proceso 1, con LST de ABI/GOES-16 no procesado en la ubicación de la estación EMA.	97
5.3. EMC de LST de ABI/GOES-16 respecto a temperatura del aire de EMAS. . . . .	98
5.4. Resultado de proceso 2, con SST de VIIRS/Suomi-NPP procesado en la ubicación de la boya meteorológica. . . . .	100
5.5. Resultado de proceso 2, con SST de VIIRS/Suomi-NPP no procesado en la ubicación de la boya meteorológica. . . . .	101
5.6. Resultado de proceso 3, con incendio activo observable en tres satélites. . . . .	103
5.7. Resultado de proceso 3, con incendio activo observable en dos satélites. . . . .	104
5.8. Resultado de proceso 3, con incendio activo observable en un satélite. . . . .	105
5.9. Resultado de proceso 3, con ningún incendio activo observable. . . . .	106





# Índice de tablas

2.1. Programas de satélites meteorológicos geostacionarios activos en 2019. . . . .	16
2.2. Programas de satélites meteorológicos de órbita polar activos en 2019. . . . .	19
2.3. Programas de satélites de observación terrestre de órbita polar activos en 2019. . . . .	21
2.4. Programas de satélites de alta resolución espacial de órbita polar activos en 2019. . . . .	23
2.5. Instrumentos del satélite GOES-16 y GOES-17. . . . .	26
2.6. Características de las bandas del sensor ABI. . . . .	27
2.7. Regiones de escaneo del sensor ABI. . . . .	27
2.8. Instrumentos del satélite Suomi-NPP. . . . .	30
2.9. Características de las bandas del sensor VIIRS. . . . .	30
2.10. Instrumento del satélite Sentinel-2. . . . .	32
2.11. Características de las bandas del sensor MSI. . . . .	33
4.1. Características de los datos satelitales utilizados. . . . .	54
4.2. Elementos del formato en el nombre estándar de los datos NetCDF4 de GOES-16. . . . .	55
4.3. Grupos y dimensiones de las variables contenidas en los archivos NetCDF4 de GOES-16. . . . .	60
4.4. Atributos del metadato de un GeoTIFF del producto SST de VIIRS/Suomi-NPP. . . . .	62
4.5. Parámetros modificables de la petición WMS del servidor de Sentinel-HUB. . . . .	63
4.6. Dimensiones de los arreglos de datos de GOES-16 y Suomi-NPP. . . . .	64

4.7. Parametros de georreferencia de los datos de GOES-16 en el formato PROJ4. . . . .	69
4.8. Sistema de referencia de los datos de GOES-16 en formatos compatibles con <i>OSR</i> . . . . .	73
4.9. Sistema de referencia para las peticiones WMS de Sentinel-2 en formato EPSG, PROJ4 y PRJ. . . .	74
4.10. Distribución de productos e intervalo de tiempo de cada satélite en los procesos. . . . .	79
4.11. Tiempo de adquisición de cada producto de acuerdo al satélite. . . . .	80
4.12. Compuestos generados por satélite y la extensión del cuadrante en el proceso 3. . . . .	90

# Capítulo 1

## Introducción

En el siguiente trabajo se describe el desarrollo de una metodología para la integración y tratamiento automático de datos satelitales en tres diferentes formatos y niveles de adquisición, provenientes de tres diferentes tipos de satélites meteorológicos y de observación terrestre: el satélite GOES-16 de la serie de satélites geoestacionarios, el satélite Suomi-NPP de la serie de satélites polares, ambos administrados por la NOAA<sup>1</sup>, y el satélite Sentinel-2 administrado por la ESA<sup>2</sup>.

Los satélites, a pesar de que han sido destinados para el monitoreo ambiental y para estudios meteorológicos, tienen características específicas distintas, como en el periodo de adquisición de datos, el tipo de sensor con el que cuentan y la órbita que recorren. Sin embargo tienen similitud en la mayoría de sus productos satelitales.

Estos productos satelitales pueden clasificarse en dos, los asociados con la obtención de variables meteorológicas, como LST (Land Surface Temperature) y SST (Sea Surface Temperature). Y los que se presentan como imágenes, las cuales permiten el monitoreo ambiental y estudios meteorológicos mediante diferentes composiciones RGB<sup>3</sup>, como TC (True Color), FT (Fire Temperature) y SWIR (Short Wave Length Infrared).

El proceso de obtención de estos productos satelitales, cambia de acuerdo a las características propias del satélite. Esto hace susceptible a que una variable o fenómeno meteorológico, sea analizado bajo la elección de un producto propio de un solo satélite.

La metodología desarrollada, permitió la integración automática de productos provenientes de los tres satélites en distintos formatos de datos y niveles de adquisición.

---

<sup>1</sup>NOAA (National Oceanic and Atmospheric Administration) es una agencia científica del Departamento de Comercio de los Estados Unidos cuyas actividades se centran en las condiciones de los océanos y la atmósfera. <https://www.noaa.gov/>

<sup>2</sup>ESA (European Space Agency) es una agencia que desarrolla la capacidad espacial de Europa. <https://www.esa.int/ESA>

<sup>3</sup>El modelo RGB se ocupa en lenguajes de programación para representar los colores, mediante tres valores colocados entre paréntesis y separados por comas.

La integración automática se realizó en un entorno de desarrollo que generalizo el tratamiento de los datos, permitiendo que la automatización y modificación de los procesos no fuera complicada. Esto se logró a partir del uso de módulos especializados para el manejo de datos espaciales, los cuales forman parte de las paqueterías y bibliotecas disponibles del lenguaje de programación Python. Adicionalmente, para la parte de adquisición y automatización se ocuparon comandos del sistema operativo GNU/Linux.<sup>4</sup>

Siguiendo esta metodología se desarrollaron distintos procesos enfocados en el monitoreo meteorológico y ambiental, llevando a cabo tareas como la extracción, tratamiento y mapeo de datos de forma automática. Estos procesos trataron de integrar las principales ventajas de los productos que se obtienen de cada tipo de satélite y sensor, además de su nivel de procesamiento previo, mostrando así un único producto conjuntado.

La versatilidad para la integración de datos, permitió en los procesos añadir información publicada de forma automática en tiempo real, tanto satelital como no satelital, proveniente de distintos servidores de datos, los cuales fueron los insumos para el desarrollo de los procesos.

La mayor parte de los scripts escritos en Python, se desarrollaron mediante funciones que se fueron reciclando en el desarrollo de cada proceso, añadiendo pequeñas modificaciones a las funciones e importándolas como módulos propios en los scripts principales.

Los procesos de integración automática que se desarrollaron fueron los siguientes:

#### **Proceso 1.**

**Datos Satelitales:** LST del sensor ABI (Advanced Baseline Imager)/GOES-16 y el compuesto RGB TC del sensor MSI (Multispectral Instrument)/Sentinel-2.

**Datos no Satelitales:** Temperatura del aire de EMAS (Estaciones Meteorológicas Automáticas).

**Producto conjuntado:** Gráfica de las últimas 24 horas de LST del sensor ABI/GOES-16 y temperatura del aire de EMAS. Incorporando la última imagen RGB TC del sensor MSI/Sentinel-2 con los datos sobrepuestos del último dato de LST del sensor ABI/GOES-16, en la ubicación de cada estación.

**Intervalo de proceso:** Cada 1 hrs.

#### **Proceso 2.**

**Datos Satelitales:** SST del sensor VIIRS(Visible Infrared Imaging Radiometer Suite)/Suomi-NPP y compuesto RGB TC del sensor MSI/Sentinel-2.

**Datos no Satelitales:** Temperatura superficial del mar de boyas meteorológicas NOAA.

---

<sup>4</sup>GNU/Linux es un sistema operativo libre tipo Unix; multiplataforma, multiusuario y multitarea, formado por el sistema GNU y el núcleo Linux en su conjunto.

**Producto conjuntado:** Grafica de las últimas 24 horas de SST del sensor ABI/GOES-16 y temperatura superficial del mar de boyas meteorológicas NOAA. Incorporando la última imagen RGB TC del sensor MSI/Sentinel-2 con los datos sobrepuestos del último dato de SST del sensor VIIRS/Suomi-NPP, en la ubicación de cada estación.

**Intervalo de proceso:** Cada 12 hrs.

### Proceso 3.

**Datos Satelitales:** Bandas del sensor ABI/GOES-16, bandas del sensor VIIRS/Suomi-NPP y compuestos RGB del sensor MSI/Sentinel-2.

**Datos no Satelitales:** Coordenadas de puntos de calor detectadas del sensor ABI/GOES-16.

**Producto conjuntado:** Cuadrantes a 200 km, 100 km y 10 km de compuestos RGB TC, FT y SWIR del sensor ABI/GOES-16, VIIRS/Suomi-NPP y MSI/Sentinel-2. Localizados en puntos de calor detectados del sensor ABI/GOES-16.

**Intervalo de proceso:** Cada 1 hrs de 10:00 a 18:00.

## 1.1. Antecedentes

En la actualidad existen gran cantidad de datos satelitales, ya que las aplicaciones no militares como el monitoreo ambiental y la meteorología así lo demandan. Además, el formato de adquisición de estos datos ha cambiado con el tiempo, ya que cada vez aumenta el tamaño de la información y se reducen los tiempos de obtención.

También la interoperabilidad entre los datos es cada vez más sencilla, ya que la mayoría de los SIG<sup>5</sup> actuales tienen como tópico hacer diferentes conjuntos de datos compatibles entre sí.[Flowerdew,1991] Esto lo logran a través de bibliotecas espaciales que conjuntan la estructura de los formatos existentes. Gracias a esto, los datos satelitales pueden tener un tratamiento similar, a pesar de provenir de distintas fuentes y estar en diferentes formatos.

Para que el tratamiento de los datos satelitales sea de manera correcta, se requiere conocer las siguientes características:

- Características del sensor y del satélite.
- Sistema de proyección y referencia.
- Formato.

---

<sup>5</sup>Integración organizada de hardware, software y datos geográficos diseñada para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada.

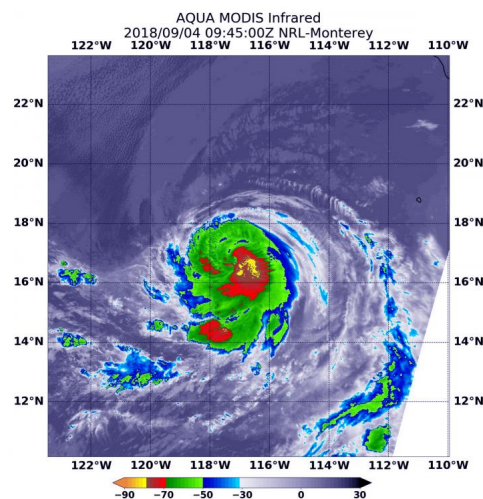
Estas tres características de manera general, dan al usuario los recursos adecuados para el tratamiento de datos satelitales.

Mucha de la información obtenida de estos datos requiere de un procesamiento automático, ya que sus principales aplicaciones se encuentran en la visualización y análisis de eventos meteorológicos y ambientales a través de imágenes o mapas.

Los mapas meteorológicos, muchas veces conocidos como mapas del tiempo, son representaciones gráficas de los valores de ciertas variables meteorológicas sobre una zona geográfica determinada, su uso ayuda a meteorólogos a realizar análisis.[Rodríguez,Benito y Portela,2004] En el caso de las imágenes, las más comunes son las que muestran el color verdadero de la Tierra, mostrando una aproximación de la visualización real de la Tierra desde el espacio.

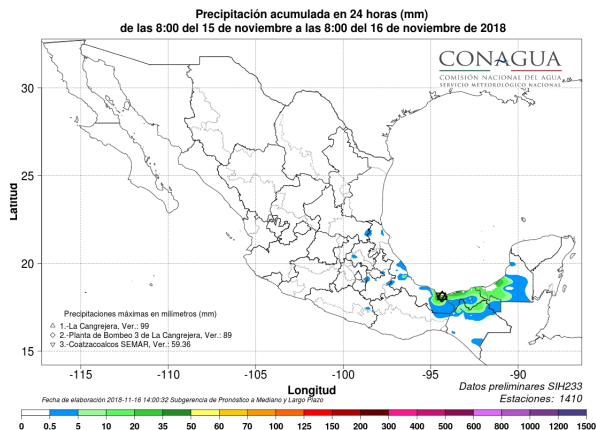
Estas imágenes y mapas generalmente son publicadas por dependencias que tienen acceso a los datos de manera directa, ya sea proveniente de antenas que reciben la señal satelital y la decodifican a valores numéricos o a través de servidores que reciben los datos a través de red.

Un ejemplo es la NOAA, la cual tiene diversas labores, como son, desarrollar pronósticos meteorológicos diarios, emitir alertas de tormentas severas, monitoreo del clima para la gestión de la pesca, la restauración costera y el apoyo al comercio marítimo[NOAA,2018]. Muchos de los resultados de estas actividades se presentan en imágenes y mapas, ver Figura 1.1, de los cuales la mayoría se generan de forma automática.



**Figura 1.1:** Imagen de temperatura de nubes del huracán Olivia, generada con datos del sensor MODIS/Aqua, publicada el 4 de septiembre del 2018 en <https://www.nhc.noaa.gov/data/>

En México un instituto similar a este es el SMN<sup>6</sup>, cuyas actividades son la vigilancia continua de la atmósfera para identificar los fenómenos meteorológicos que pueden afectar las distintas actividades económicas y originar la pérdida de vidas humanas.[SMN,2017] Por lo cual, algunos de sus resultados los presenta en forma de imágenes y mapas, generados también de forma automática, ver Figura 1.2.



**Figura 1.2:** Mapa de precipitación acumulada de 1410 estaciones del SMN, publicada el 16 de noviembre del 2018 en <https://smn.cna.gob.mx/es/climatologia/temperaturas-y-lluvias/mapas-diarios-de-temperatura-y-lluvia>

Las dependencias antes mencionadas y otras similares, utilizan módulos para el manejo de datos espaciales que facilitan el procesamiento automático de los datos satelitales. Existen diversos paquetes y softwares que contienen estos módulos, tanto de licencia Open Source<sup>7</sup> como privados.

Los módulos se clasifican de acuerdo a su función en cuatro grandes grupos:

- Acceso de datos.
- Geometría computacional y procesamiento numérico.
- Herramientas o software de escritorio.
- Visualización de datos.

### Acceso de datos.

Son los que se encargan de leer, escribir y transformar datos espaciales, ya que los conjuntos de datos son grandes, complejos y variados[Lawhead,2015], lo cual provoca que la interoperabilidad entre datos se dificulte. Estos módulos son llamados más comúnmente bibliotecas espaciales, ya que contienen los manejadores adecuados para cada

<sup>6</sup>SMN (Servicio Meteorológico Nacional) es el organismo público de México encargado de proporcionar información sobre el estado del tiempo que prevalece en el país.

<sup>7</sup>Es un modelo de desarrollo de software basado en la colaboración abierta.



formato, tanto tipos de datos raster<sup>8</sup> como vectorial<sup>9</sup>.

GDAL(Geospatial Data Abstraction Library) es por mucho la más usada a nivel mundial, alrededor de 100 programas de SIG la utilizan en sus procesos, pues soporta casi todos los formatos espaciales, más de 140 formatos raster y más de 80 formatos vectoriales.[GDAL,2018] Anteriormente la biblioteca OGR se encargaba de la parte vectorial y GDAL de la raster, pero desde la versión 2.0 de GDAL, ambos se conjuntaron.

### **Geometría computacional y procesamiento numérico.**

La geometría computacional abarca los algoritmos necesarios para realizar operaciones con datos vectoriales. [Lawhead,2015] Los enfocados en sistemas de proyecciones y sistemas referencias, contienen bibliotecas de códigos y algoritmos de transformación entre coordenadas como PROJ.4.

Los enfocados a procesamiento geométrico computacional contienen los algoritmos necesarios para realizar operaciones con datos vectoriales<sup>10</sup>, los más usados son JTS, GEOS y CGAL.

Los que trabajan con bases de datos y permiten realizar operaciones vectoriales y raster, están relacionados con las herramientas de los manejadores de datos espaciales, como PostGIS, Oracle-Spatial, ArcSDE y MySQL.[Lawhead,2015]

El procesamiento matricial abarca módulos para el manejo numérico de datos raster. Unos de los más usados son GDAL y GMT.

GDAL además de ser una biblioteca espacial, también incluye un conjunto de herramientas de línea de comandos que pueden realizar una variedad de operaciones sin ninguna programación. Permite realizar análisis, conversiones, extracción, re proyecciones, operaciones matemáticas, entre otros.

GMT (Generic Mapping Tools) es un conjunto de programas desarrollado para Linux, también trasladado para Windows, y que se ejecuta en una ventana de comandos o terminal. Puede crear mapas de situación, manipular datos, hacer operaciones matemáticas sobre mallas, mapas de contorno, sombreados de relieve, mapas vectoriales, filtrados, ajuste de tendencias, mallados a partir de datos discretos, entre otras cosas.[Alvarez,2012]

### **Herramientas o softwares de escritorio.**

La mayoría de los SIG de escritorio permite realizar procesamientos de datos espaciales de forma automatizada, permitiendo crear scripts a través de un lenguaje de programación, usando las mismas funciones que en la versión de escritorio. Los más usados son los módulos en lenguaje Python de ArcGIS, QGIS, GRASS y gvSIG.

Otros softwares utilizan otro tipo de lenguajes de programación, como ENVI, el cual utiliza IDL. IDL es usado por diferentes disciplinas para el manejo y representación de complejos datos numéricos. Puede interpretar datos,

<sup>8</sup>Matriz de celdas (o píxeles) organizadas en filas y columnas, donde cada celda contiene un valor.

<sup>9</sup>Representación espacial es mediante puntos, líneas y polígonos.

<sup>10</sup>Procesos como calculo de zonas de influencia, operaciones de intersección, unión, diferencia, recorte, entre otros.

automatizar procesos y crear poderosas aplicaciones.[SIGSA,2018]

Otro software muy usado es MatLab, el cual contiene el paquete Mapping ToolBox. Este tiene la capacidad de importar un alta gamma de formatos y archivos espaciales, permitiendo tratarlos en el entorno de MatLab. También permite la visualización y creación de mapas en 2D y 3D, otorgando una alta customización en cuanto a proyecciones, propiedades del mapa, símbolos, paletas de colores, entre otros.

### **Visualización de datos.**

En los módulos para visualización de datos espaciales predominan las aplicaciones Web SIG, ya que la descarga y visualización de datos es considerada su función principal y no el tratamiento de datos, del cual se encuentran muy limitados. La mayoría de estas aplicaciones son administradas a partir de una base de datos espacial como PostGIS, un servidor de datos como Geoserver y una aplicación web desarrollada con una biblioteca JavaScript<sup>11</sup> como OpenLayers o Leaflet.

En estas cuatro categorías están distribuidos los principales módulos que permiten la automatización de procesamiento de datos satelitales, donde predominan más los formatos raster y multidimensional que los vectoriales.

Si bien la mayoría de los módulos en cada categoría están ligados a diversos lenguajes de programación, existen dos lenguajes que han incluido en sus propios repositorios de paquetes la mayoría de los módulos para manejo de datos espaciales existentes, muchos son los mismos módulos pero adaptados a cada lenguaje. Por lo cual la mayor parte de las cuatro categorías mencionadas anteriormente, puede realizarse desarrollando scripts en un solo lenguaje de programación. Estos dos lenguajes son:

- R.
  
- Python.

### **R.**

R es una plataforma y lenguaje de análisis estadístico de licencia Open Source con herramientas gráficas muy avanzadas, es un referente en el análisis estadístico desde hace muchos años. Permite la importación de módulos al inicio del script, en los cuales se pueden incluir funciones para leer, visualizar y analizar datos espaciales. Permite usar las funciones base de R y complementarlas con paquetes contribuidos a CRAN<sup>12</sup>, lo que permite hacer mapas añadiendo información vectorial, raster, multidimensional<sup>13</sup> e incluso añadir información de un servidor de mapas web como Leaflet y GoogleMaps.

---

<sup>11</sup>Código pre-escrito en JavaScript que permite un desarrollo más fácil de aplicaciones.

<sup>12</sup>Red de servidores FTP y web de todo el mundo que almacena versiones idénticas y actualizadas de código y documentación para R.

<sup>13</sup>Contienen una o múltiples variables y cada variable es un conjunto multidimensional que representa datos de una determinada dimensión vertical, los mas comunes son NetCDF, GRIB y HDF.

## Python.

Python es un lenguaje de programación de alto nivel, esto quiere decir que puede ser utilizado para crear pequeñas aplicaciones o grandes programas. Es un lenguaje interpretado y multiparadigma, ya que soporta programación orientada a objetos, programación imperativa y programación funcional.[Python,2018] Su código es similar al pseudocódigo<sup>14</sup>, por lo que lo convierte en uno de los lenguajes más sencillos de usar.

Es de licencia Open Source, por lo cual la comunidad de desarrolladores ha creado un gran número de módulos, muchos para el manejo de datos, visualización, cálculo simbólico y aplicaciones científicas específicas. Esto lo hace capaz de procesar datos espaciales de manera muy sencilla, y crear procesos complejos con poco código.

Si bien los dos son lenguajes Open Source y comparten algunos módulos para el manejo de datos espaciales, es Python el que contiene la mayor cantidad de ellos. Ya que se encuentran prácticamente todos los módulos mencionados en las cuatro categorías anteriores, siendo los mismos módulos o adaptaciones de ellos. Es por ello que los SIG más populares como ArcGIS, QGIS, GRASS, entre otros, han adoptado a Python como el lenguaje oficial para el uso de sus herramientas a través de scripts.

De esta misma forma, en este trabajo se eligió a Python como la plataforma de integración de datos satelitales, ya que si un solo lenguaje de programación cuya disponibilidad de módulos espaciales permite abarcar las cuatro categorías mencionadas anteriormente, puede por lo tanto conjuntar datos satelitales en diferentes formatos, provenientes de diversos productos satelitales, que a su vez provienen de tres diferentes tipos de satélites, todo de forma automática.

## 1.2. Planteamiento del problema

La interoperabilidad de datos provenientes de satélites de observación terrestre y meteorológica se ha mejorado en gran medida en los últimos años, debido al constante desarrollo de módulos para manejo de datos espaciales. Aun así es común ver que un fenómeno meteorológico y ambiental se analiza con ayuda de datos de un solo satélite, ya que las características de observación que tienen, los catalogan de cierta forma para un uso específico. Sin embargo, muchos productos satelitales provenientes de diferentes satélites, son equivalentes en el nombre y similares en su obtención.

Si los módulos existentes permiten la interoperabilidad de datos, aunque se encuentren en diferentes formatos, se pueden crear procesos que integren la información satelital de forma automática para el análisis de un mismo fenómeno, obteniendo así más recursos para su análisis.

---

<sup>14</sup>Descripción compacta e informal, del principio operativo de un programa informático u otro algoritmo.

### 1.3. Objetivos

- Desarrollar una metodología mediante el uso de módulos para manejo de datos espaciales, disponibles en el lenguaje Python, para la integración de datos provenientes de tres diferentes satélites meteorológicos y de observación terrestre, en tres diferentes formatos de adquisición.
- Desarrollar procesos para el estudio meteorológico y de observación terrestre que se basen en la metodología descrita.
- Automatizar los procesos en un servidor con sistema operativo GNU/Linux.



## Capítulo 2

# Observación meteorológica satelital

Como se mencionó anteriormente, el tratamiento correcto de los datos satelitales requiere del conocimiento previo acerca de las características físicas que definen al satélite, así como del sensor que llevan incorporado y que registró los datos.

Además los satélites meteorológicos tienen en común diversos conceptos y características generales que los diferencian de otro tipo de satélites artificiales, como los de comunicaciones o los de navegación.

### 2.1. Satélites meteorológicos

Un satélite meteorológico es aquel que tiene como principal propósito el monitoreo de la superficie terrestre y su atmósfera. De esto se derivan algunos procesos y técnicas para la obtención de productos a partir de los datos satelitales que registra.

Una aplicación de estos productos son suministrar información sobre la cobertura nubosa, la cual permite identificar y seguir de cerca en la escala global diferentes eventos, como los sistemas de baja presión atmosférica, los sistemas frontales<sup>1</sup>, las corrientes de chorro<sup>2</sup>, la ubicación de convergencia intertropical<sup>3</sup> y los ciclones tropicales.[González,2010] También suministran información para determinar el tiempo atmosférico y el clima.

Además, pueden captar información sobre el medio ambiente como incendios, contaminación, tormentas de polvo y arena, concentración de clorofila, actividades volcánicas, estado de la vegetación, corrientes oceánicas, luces de ciudad, concentración de sedimentos, humo, auroras boreales, entre otras. [Satélite meteorológico,2018]

---

<sup>1</sup>Espacio entre dos masas de aire de distintas características termodinámicas.

<sup>2</sup>Flujo de aire rápido y estrecho que se encuentra en la atmósfera.

<sup>3</sup>Zona donde convergen los vientos alisios del hemisferio norte con los del hemisferio sur.

Algunos productos que derivan de instrumentos más especializados permiten obtener otro tipo información, como la medición del campo gravitatorio de la Tierra, el nivel medio del mar, el grosor de los glaciares, la presencia de rayos, el balance radioactivo, la humedad del suelo, la salinidad e incluso hacer mediciones de monitoreo al sol.

Los instrumentos que permiten obtener esta información, funcionan a partir de sensores que transforman la radiación electromagnética<sup>4</sup> en información perceptible y analizable<sup>5</sup>, los sensores pueden clasificarse en dos tipos:

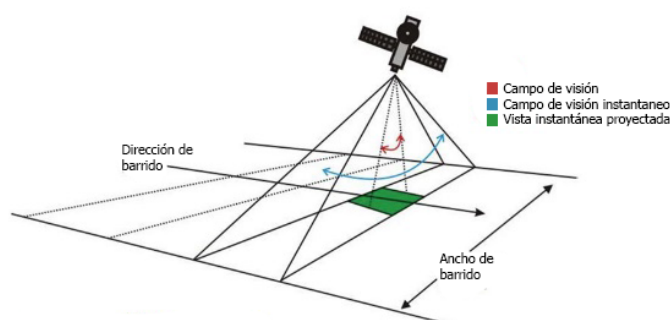
- Sensores pasivos.
- Sensores activos.

### Sensores pasivos.

Estos detectan la radiación electromagnética emitida o reflejada de fuentes naturales o artificiales. [Chuvieco,1995]

Tienen dos características espaciales, ver Figura 2.1:

- **Campo de vista instantáneo:** Es el ángulo en que el detector es sensible a la radiación y determina en conjunto con la altura del sensor, el tamaño del elemento más pequeño de la imagen (pixel).[Chuvieco,1995]
- **Anchura de barrido:** Es la distancia lineal en ángulo recto que va rastreando el barredor<sup>6</sup> en tierra, está en función del campo de visión.[Chuvieco,1995]



**Figura 2.1:** Elementos espaciales básicos de los sensores pasivos, modificado de <https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20180000780.pdf>.

<sup>4</sup>Combinación de campos eléctricos y magnéticos, que se propagan a través del espacio en forma de ondas portadoras de energía.

<sup>5</sup>Datos espaciales vectoriales, raster o multidimensionales.

<sup>6</sup>Sistema óptico pendular o rotatorio que en cada ciclo capta energía proveniente de una franja de la superficie.

Los sensores pasivos, de acuerdo con su tipo de funcionamiento, pueden agruparse como:

- Sistemas fotográficos.
- Óptico electrónicos<sup>7</sup>.
- Espectrómetros<sup>8</sup> de imagen.
- Radiómetros<sup>9</sup> de microondas<sup>10</sup>.

De acuerdo con la forma de captura, los sensores activos pueden agruparse en:

- Radiómetros de barrido o de barredor mecánico.
- Radiómetros de empuje.

### **Sensores activos.**

Tienen la capacidad de emitir un haz energético, que posteriormente, captan tras su reflexión sobre la superficie observada.[Chuvieco,1995]

De acuerdo con su tipo de funcionamiento, pueden agruparse como:

- Ecosonda y sonares.
- Radares<sup>11</sup>.
- LIDAR<sup>12</sup>.

La radiación captada por los sensores es un fenómeno continuo en 4 dimensiones (espacio, tiempo, longitud de onda y radiancia), por lo que los datos son caracterizados con 4 tipos de resoluciones:

- Resolución espacial.
- Resolución espectral.
- Resolución radiométrica.

---

<sup>7</sup>Óptica similar a la fotográfica con un sistema de detección electrónica

<sup>8</sup>Instrumento que sirve para medir las propiedades de la luz en una determinada porción del espectro electromagnético.

<sup>9</sup>Sistema óptico pendular o rotatorio que en cada ciclo capta energía proveniente de una franja de la superficie.

<sup>10</sup>Longitud de onda entre 1 milímetro y un metro, su frecuencia es de entre 300 MHz y 300 GHz.

<sup>11</sup>Sistema que usa ondas de radio para medir distancias, altitudes, direcciones y velocidades.

<sup>12</sup>Dispositivo que permite determinar la distancia desde un emisor láser a un objeto o superficie utilizando un haz láser pulsado.



- Resolución temporal.

### **Resolución espacial.**

Esta referida al tamaño de la unidad mínima de los datos, si son imágenes se refiere al tamaño del pixel. Las unidades de medida son determinadas de acuerdo con el sistema de coordenadas de los datos. La resolución espacial depende de la altura del sensor, el ángulo de visión, la velocidad de escaneado y características ópticas del sensor.[Chuvieco,1995]

### **Resolución espectral.**

Esta referida al número de bandas<sup>13</sup> espectrales que puede captar el sensor.[Chuvieco,1995]

### **Resolución radiométrica.**

Esta referida a la cantidad de niveles digitales<sup>14</sup> que describen los datos de radiancias del sensor. Es representado en una base binaria 2n.[Chuvieco,1995]

### **Resolución temporal.**

Esta referido a la frecuencia a la que pasa el sensor en el mismo punto, depende de las características de la órbita del satélite.[Chuvieco,1995]

La clasificación más importante de los satélites meteorológicos y de observación terrestre está en función del tipo de órbita que recorren. Existen dos tipos de orbita:

- Órbita Geoestacionaria.
- Órbita Polar.

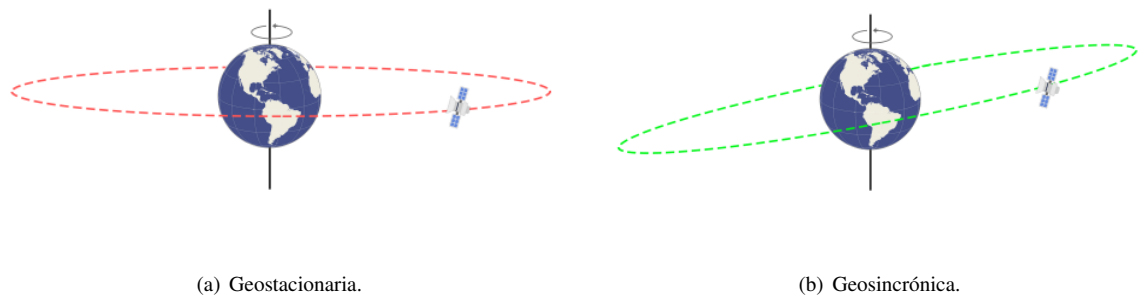
### **Órbita Geoestacionaria.**

La órbita geoestacionaria es un caso de la órbita geosincrónica, ya que la función de ambas consiste en que el satélite este en sincronía con la velocidad de rotación de la Tierra (1670 km/h en el ecuador), lo que hace que el satélite permanezca siempre vertical en una misma coordenada y observe siempre a la misma región. La diferencia entre entre las dos, es que la orbita geosincrónica pueden tener cualquier inclinación, ver Figura 2.2.

Los satélites de órbita geoestacionaria, siempre están posicionados en el ecuador y a una altura orbital de 36000 Km aproximadamente. Su velocidad a esa altura debe de ser de unos 11070 km/h para mantener el equilibrio entre la fuerza centrífuga y gravitacional, lo que le permite estar en órbita.[Luque y Amengual,n.f]

<sup>13</sup>Información en rangos definidos del espectro electromagnético.

<sup>14</sup>La señal captada por el sensor, transmitida electrónicamente en forma digital como una serie de números.



**Figura 2.2:** Diferencia entre órbita Geoestacionaria (a) y órbita Geosincrónica (b), modificados de <https://airfreshener.club/quotes/orbit-ground-geostationary-geosynchronous-vs-traces.html>

Una de las principales ventajas de ello, es que la señal recibida del satélite puede ser captada por una antena terrestre fijada permanentemente en la posición del satélite. Permitiendo que los datos tengan una alta resolución temporal, haciéndolos adecuados para el monitoreo meteorológico continuo.

La gran altitud de los satélites y el gran ángulo de visión del sensor, junto con la rápida velocidad de escaneado que poseen, hacen que la resolución espacial de los datos sea baja.

Los programas activos de satélites meteorológicos geoestacionarios cubren la mayor parte de la superficie terrestre, exceptuando los polos.

Los programas activos se encuentran en la Tabla 2.1:

Tabla 2.1: Programas de satélites meteorológicos geoestacionarios activos en 2019 (satélite utilizado en color gris), fuente de NOAA, ESA, JMA, Roskosmos, ISRO y NSMC.

Programa	Nombre	Nación	Desarrollo/ Administración	Satélite operativo	Inicio operación	Duración	Longitud	Altitud	Sustituto/Año
GOES EAST	Geostationary Operational Environmental Satellite	EUA	NASA NOAA	GOES-R o GOES-16	18-Dic-2017	15 años	72.5 ° W	Perigeo 35,780.2 km Apogeo 35,793.1 km	GOES-T/2020
GOES WEST	Geostationary Operational Environmental Satellite	EUA	NASA NOAA	GOES-S o GOES-17	12-Feb-2019	15 años	137.2 ° W	Perigeo 35,773.1 km Apogeo 35,799.1 km	GOES-U/2024
METEOSAT	-	Unión Europea	ESA EUMESAT	MSG-4 o Meteosat-11	16-Dic-2015	10 años	0°	Perigeo 35,787.4 km Apogeo 35,798.5 km	MTG-11 o Meteosat-12/2019
GMS o Himawari	Geostationary Meteorological Satellite	Japón	Mitsubishi Electric-Boeing/ JMA	Himawari 8	7-Jul-2015	8 años	140.7 ° E	Perigeo 35,791 km Apogeo 35,795 km	Himawari-9/2022
GOMS	Geostationary Operational Meteorological Satellite	Rusia	NPO Lavochkin/ Roskosmos	GOMS-2 o Electro-L No.2	21-Ene-2016	10 años	77.8° E	Perigeo 35,791.1 km Apogeo 35,797.1 km	GOMS-3 o Electro-L No.3/2019
INSAT	Indian National Satellite System	India	ISRO	INSAT-3DR	17-Sep-2016	10 años	74° E	Perigeo 35,731 km Apogeo 35,841 km	INSAT-3DS/2022
FengYun	-	China	SAST/NSMC	FY-2G o Feng-Yun-2G	3-Jun-2015	4 años	105° E	Perigeo 35,781.6 km Apogeo 35,811.8 km	FY-2H o Feng-Yun-2H/2018

### Órbita Polar.

La función de la órbita polar consiste en recorrer la Tierra de polo a polo, la altitud con la que recorren los satélites esta órbita oscila entre los 400 y 1500 km.

La órbita polar se puede describir como un plano meridiano<sup>15</sup> ligeramente inclinado respecto al eje de rotación de la Tierra, esta inclinación es necesaria por la forma elipsoidal de la Tierra, que hace variar la trayectoria del satélite y también porque la mayoría de las órbitas polares buscan ser heliosincrónicas [Luque y Amengual, n.f], ver Figura 2.3.

Una órbita heliosincrónica busca tener una posición fija determinada a un mismo tiempo solar durante todos sus pasos, esto quiere decir que pasara el satélite siempre a la misma hora local en cada punto. Las órbitas polares heliosincrónicas aprovechan esta cualidad para aplicaciones meteorológicas, de observación y monitoreo terrestre.



**Figura 2.3:** Trayectoria de la órbita polar, modificado de <https://airfreshener.club/quotes/orbit-ground-geostationary-geosynchronous-vs-traces.html>

No todas las órbitas heliosincrónicas son polares, estas pueden ser también circulares y cumplir con el mismo objetivo, sin embargo las órbitas circulares tienen la desventaja de no monitorear todas las latitudes.

También hay órbitas polares sincrónicas al día y noche, obteniendo dos pasos en 24 horas a las mismas dos horas locales. Esta capacidad dependerá del tiempo que le tome al satélite recorrer la órbita, el cual a su vez depende de la altitud e inclinación a la que esté el satélite. En altitudes de 850 a 1000 km y con inclinación cerca de  $98^\circ$  (siendo  $90^\circ$  una orbital polar y  $0^\circ$  ecuatorial), suele ser de 90 a 100 min, con lo cual recorren de 14 a 15 veces la Tierra diariamente. [Órbita heliosíncrona, 2018]

Los datos de los satélites polares al tener órbitas más bajas, tienen una resolución espacial muy alta, en consecuencia a esto, el tiempo que les toma en volver a pasar por el mismo punto es más alto, provocando que tengan una resolución temporal más baja frente a los satélites geoestacionarios.

<sup>15</sup>Es el plano definido por la vertical del lugar y el eje de la Tierra.

El hecho de que pasen al mismo tiempo local, otorga datos de radiación muy uniformes, por lo cual se han asignado para realizar ciertas operaciones de observación y monitoreo terrestre, que a su vez permiten ver cambios más detallados en la superficie.

A diferencia de los geoestacionarios, la señal recibida de los satélites de órbita polar, debe ser captada por una antena mecánica que sigue la predicción del paso de un satélite, mediante parámetros de órbita contenidos en formatos como el TLE.<sup>16</sup>

Los satélites meteorológicos de órbita polar, a diferencia de los geoestacionarios, pueden clasificarse de acuerdo a la función que desempeñan, ya que estos fueron desarrollados más para la observación y monitoreo terrestre, que para la meteorología, a diferencia de los geoestacionarios que son casi exclusivos de uso meteorológico.

Dentro de los de órbita polar, se pueden distinguir tres grupos:

- Satélites meteorológicos.
- Satélites de observación terrestre.
- Satélites de alta resolución espacial.

### **Satélites meteorológicos.**

Su altitud oscila entre los 850 y 1500 km, sus órbitas polares pueden ser heliosincrónicas, no sincrónicas y circulares.

Los programas activos más importantes en 2019 se encuentran en la Tabla 2.2.

---

<sup>16</sup>Formato de datos que codifican una lista de elementos orbitales de un objeto orbital de la Tierra para un momento dado.

**Tabla 2.2:** Programas de satélites meteorológicos de órbita polar activos en 2019 (satélite utilizado en color gris), fuente de NOAA, ESA, NSMC y JAXA.

Programa	Nombre	Nación	Desarrollo/ Administración	Satélite operativo	Inicio operación	Duración	Temporalidad	Altitud	Sustituto/Año
JPSS-1	Joint Polar Satellite System	EUA	NASA/ NOAA	NOAA-20o JPSS-1	30-May-2018	7 años	12 hrs	Perigeo 824.3 km Apogeo 827.8 km	NOAA-21/2022
SNPP	Suomi National Polar-orbiting Partnership	EUA	NASA/ NOAA	Suomi-NPP	5-Ene-2014	5 años	12 hrs	Perigeo 833.7 km Apogeo 834.3 km	NOAA-20/2018
Jason	Joint Altimetry Satellite Oceanography Network	EUA/ Europa	NASA/ NOAA - CNES - EUMETSAT	Jason-3	21-Jun-2016	5 años	9.9 días	Perigeo 1331.7 km Apogeo 1343.7 km	Jason-CS/2020
MetOp	-	Europa	EUMETSAT	MetOp-B	24-Abr-2018	4.5 años	12 hrs	Perigeo 819 km Apogeo 821 km	MetOp-C/2018
Fengyun	-	China	SAST/ NSMC	FY-3DoFeng-Yun-3D	14-Nov-2017	5 años	12 hrs	Perigeo 832.6 km Apogeo 834.4 km	FY-3E oFeng-Yun-3E/2022
GCOM	Global Change Observation Mission	Japon	JAXA	GCOM-CI	23-Dic-2017	5 años	2 días	Perigeo 788 km Apogeo 806 km	-

### **Satélites de observación terrestre.**

Las órbitas polares de los satélites de observación terrestre suelen ser heliosincronicas. También son conocidos como satélites de recursos naturales, ya que se emplean para monitorear las condiciones de la superficie terrestre a partir de imágenes en una resolución más elevada que los meteorológicos.

Cubren áreas más pequeñas de escaneo para obtener una mayor resolución espacial en las imágenes, lo que provoca que tarden más tiempo en cubrir toda la Tierra. La resolución espectral con la que cuentan, los posibilita para distinguir superficies y cartografiar.[Eduspace,2011]

Los programas activos más importantes se encuentran en la Tabla 2.3.

**Tabla 2.3:** Programas de satélites de observación terrestre de órbita polar activos en 2019 (satélite utilizado en color gris), fuente de USGS, ESA, INPE, e ISRO.

Programa	Nombre	Nación	Desarrollo/ Administración	Satélite operativo	Inicio operación	Duración	Temporalidad	Altitud	Sustituto/ Año
LANDSAT	-	USA	NASA/USGS	Landsat-8	11-Abr-2013	< 5 años	16 días	Perigeo 708.7 km Apogeo 710.6 km	Landsat-9/ 2020
Copernicus	-	Europa	ESA/CE	Sentinel-2A	23-Jun-2015	< 5 años	10 días	Perigeo 795.2 km Apogeo 797.1 km	Sentinel-2C/ 2022
Copernicus	-	Europa	ESA/CE	Sentinel-2B	7-Mar-17	< 5 años	10 días	Perigeo 778.25 km Apogeo 780.84 km	Sentinel-2D/ 2023
Copernicus	-	Europa	ESA/CE	Sentinel-1A	3-Abr-2014	< 5 años	10 días	Perigeo 695 km Apogeo 696 km	Sentinel-1C/ 2022
Copernicus	-	Europa	ESA/CE	Sentinel-1B	25-Abr-2016	< 5 años	10 días	Perigeo 695 km Apogeo 696 km	Sentinel-1D/ 2023
CBERS	China-Brazil Earth Resources Satellite	China/ Brasil	CNSA/INPE	CBERS-4 o Ziyuan 1-04	7-Dic-2014	3 años	26 días	Perigeo 779 km Apogeo 781 km	CBERS-4A/ 2019
Resourcesat	-	India	ISRO	Resourcesat-2A	7-Dic-2016	5 años	24 días	Perigeo 813 km Apogeo 825 km	ResourceSat-3/ 2019



### **Satélites de alta resolución espacial.**

Los satélites con una resolución menor o igual a 5 metros se consideran de alta resolución espacial. Sus orbitas polares suelen ser heliosincronicas.

Debido a su estructura de alta resolución, suelen estar limitados en la resolución espectral y en los instrumentos que llevan, usualmente sensores pancromáticos<sup>17</sup> y multiespectrales. Pueden llegar a obtener resoluciones espaciales de 30 cm.

Las imágenes de alta resolución son generalmente de nivel comercial y son utilizadas para la detección de cambios, estereoscopia<sup>18</sup> en 3D y cartografía de alta resolución.

Los programas activos más importantes se encuentran en la Tabla 2.4.

---

<sup>17</sup>Imagen fotográfica sensible a todas las longitudes de onda del espectro visible.

<sup>18</sup>Técnica capaz de recoger información visual tridimensional y/o crear la ilusión de profundidad mediante una imagen.

**Tabla 2.4:** Programas de satélites de alta resolución espacial de órbita polar activos en 2019, fuente de DigitalGlobe, ADS, KARI y EADS.

Programa	Nombre	Nación	Desarrollo/ Administración	Satélite operativo	Inicio operación	Duración	Temporalidad	Altitud	Sustituto/ Año
WorldView	-	EUA	DigitalGlobe	WorldView-4	11-Nov-2016	7 años	1 día	Perigeo 612 km Apogeo 615 km	-
GeoEye	-	EUA	DigitalGlobe	GeoEye-1	6-Oct-2008	10 años	3 día	Perigeo 673 km Apogeo 685 km	WorldView-4/ 2018
Pleiades	-	Francia	CNES / Airbus Defence and Space	Pleiades-1A	17-Dic-2011	7 años	26 días	Perigeo 697 km Apogeo 699 km	-
Pleiades	-	Francia	CNES / Airbus Defence and Space	Pleiades-1B	17-Dic-2011	7 años	26 días	Perigeo 697 km Apogeo 699 km	-
Kompsat	-	Corea	KARI	Kompsat-3A	25-Mar-2015	4 años	28 días	Perigeo 530.4 km Apogeo 543.5 km	-
SPOT	Satellite Pour l'Observation de la Terre	Francia/ Italia	EADS Astrium	Spot-7	5-Ago-2014	10 años	26 días	Perigeo 703.9 km Apogeo 705.9 km	-

En este trabajo debido a las condiciones de adquisición automática con las que se contó y además del estado operativo actual, se utilizaron datos de los siguientes satélites:

#### **Satélites geoestacionarios:**

- GOES-16.

#### **Satélites de órbita polar.**

- Meteorológicos.
  - Suomi-NPP.
- Observación terrestre.
  - Sentinel-2.

Se describirán de forma más detallada a continuación.

## **2.2. Satélites geoestacionarios**

### **2.2.1. GOES-16**

El satélite GOES-16 pertenece a la tercera generación de satélites geoestacionarios desarrollados por la NASA y administrados por la NOAA. El DOC<sup>19</sup> es el principal financiador del proyecto y la autoridad de aprobación de presupuesto.[Miller,2003]

Continúa con la serie de satélites del programa GOES que inicio en 1974, proporcionando datos atmosféricos, hidrológicos, oceánicos, climáticos, solares y espaciales.[GOES-R-eoPortal,n.f] El satélite GOES-16 tiene como objetivos ofrecer capacidades de observación mejoradas en las cuatro resoluciones (espacial, temporal, espectral y radiométrica) que sus predecesores.

Está programado para operar por un período de al menos 15 años, siendo sustituido por el satélite GOES-T lanzado en 2020 que ampliarán la disponibilidad de datos de la serie hasta 2036, siendo renombrado una vez que este en estado operacional como GOES-18[GOES-R-eoPortal,n.f], ver Figura 2.4.

El satélite GOES-16 era nombrado GOES-R en su fase de pre lanzamiento, cuando llego a orbita fue renombrado a GOES-16. Este fue lanzado el 19 de Noviembre del 2016, siendo operativo hasta el 18 de Diciembre del 2017,

<sup>19</sup>Departamento del gobierno de los Estados Unidos dedicado a la promoción del crecimiento económico del país.

cuando alcanzo su posición operacional en 75.2 ° W. Por su localización es también llamado GOES EAST. El satélite GOES-17 se encuentra en la posición 137.2° W y es llamado GOES WEST, el cual en su fase de pre lanzamiento se le nombro GOES-S. Fue lanzado el 1 de Marzo del 2018 y es operativo desde Enero del 2019.[GOES-R-eoPortal,n.f]

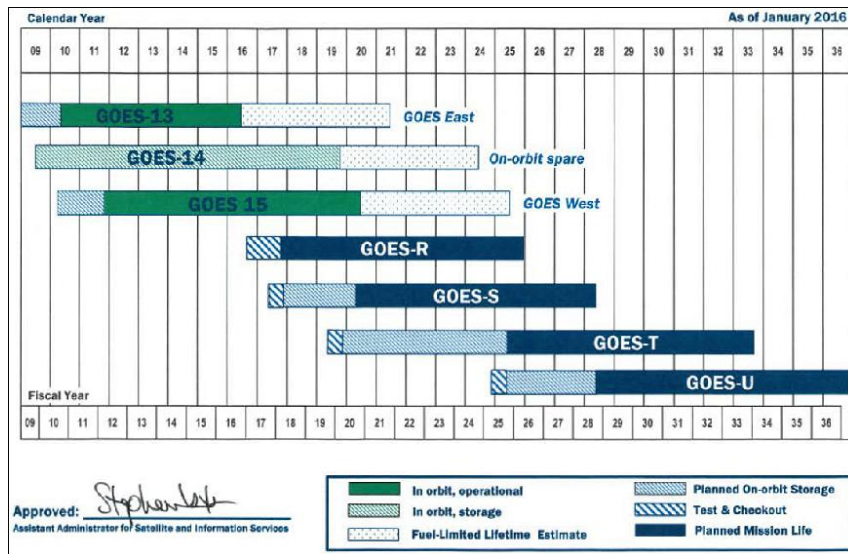


Figura 2.4: Calendario oficial de próximos satélites del programa GOES de la NOAA aprobado hasta 2036, tomado de <https://www.nesdis.noaa.gov/GOES-R-Series-Satellites>.

El satélite GOES-16 se encuentra en una altitud promedio de 35786 km.

El programa GOES, está compuesto de dos segmentos, el espacial y el terrestre, siendo el primero los dos satélites GOES (16 y 17) y el segmento terrestre un sistema integrado para proporcionar capacidades de procesamiento, control y monitoreo de los datos[Royle,2011], ver Figura 2.5.

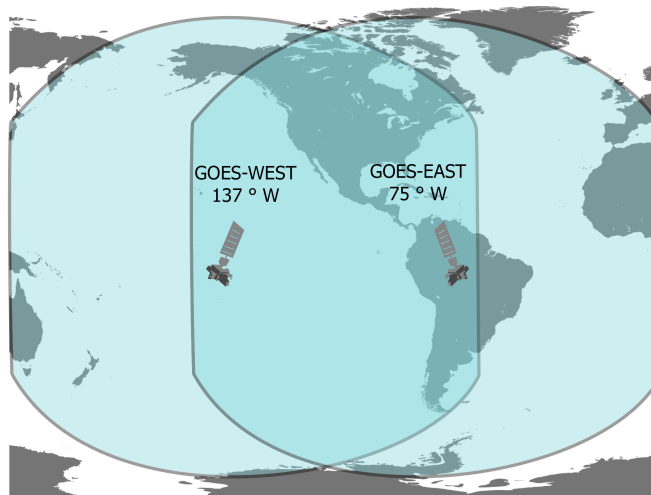


Figura 2.5: Cobertura del segmento espacial del programa GOES, con los satélites GOES-WEST y GOES-EAST posicionado en 137° W y 75° W respectivamente, modificado de <https://www.nesdis.noaa.gov/GOES-R-Series-Satellites>.

El segmento espacial, comparte las mismas características en arquitectura satelital y los mismos instrumentos integrados, ver Tabla 2.5.

**Tabla 2.5:** Instrumentos del satélite GOES-16 y GOES-17 (sensor utilizado en color gris), fuente de NOAA.

Instrumentos	Nombre	Descripción
<b>Orientación Terrestre.</b>		
ABI	Advanced Baseline Imager	Es un radiómetro de imagen pasiva multicanal, 2 bandas en el visible, 4 en IR y 10 en IR lejano. Es el principal instrumento del GOES, generando el 65% de los productos.
GLM	Geostationary Lightning Mapper	Es un detector del infrarrojo cercano en un solo canal, capaz de captar la luz de corta duración emitida por un rayo. Detecta en un 79 a 90% la totalidad de rayos. Es el primero en un satélite geostacionario.
<b>Orientación Solar</b>		
EXIS	Extreme Ultraviolet and X-ray Irradiance Sensors	Par de sensores (EVUS y XRS) que monitorean la irradiación solar en la atmosfera. Detecta llamaradas y erupciones solares.
SUVI	Solar Ultraviolet Imager	Es un sensor ultravioleta de 6 bandas que produce imágenes del sol, que sirven para detectar características solares como agujeros y eyecciones coronales.
<b>Entorno espacial</b>		
MAG	Magnetometer	Magnetómetro que mide el campo magnético de la Tierra en la extensión de la magnetosfera, desde la órbita geostacionaria.
SEISS	Space Environment In-Situ Suite	4 sensores que monitorean el flujo de protones, electrones e iones en la magnetosfera.

En este trabajo se utilizaron exclusivamente datos del sensor ABI, ya que los productos utilizados se derivan de sus bandas. Las bandas del sensor ABI tienen distintas aplicaciones, así como diferentes resoluciones espaciales, ver Tabla 2.6.

**Tabla 2.6:** Características de las bandas del sensor ABI (bandas utilizadas en color gris), fuente de NOAA.

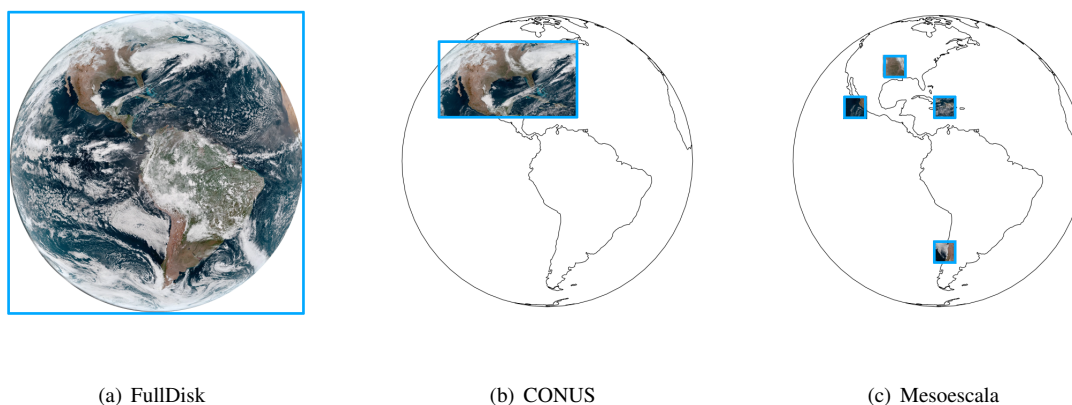
Canal	Longitud de onda ( $\mu\text{m}$ )	Longitud de onda central ( $\mu\text{m}$ )	Resolución espacial (km)	Objetivos de medición y uso de datos
1	0.45-0.49	0.47	1	Aerosol diurno sobre tierra, cartografía de aguas costeras
2	0.59-0.69	0.64	0.5	Nubes de día niebla, insolación, vientos.
3	0.846-0.885	0.885	1	Vegetación diurna / áreas quemadas y aerosoles sobre el agua, vientos
4	1.371-1.386	1.378	2	Nube cirrus durante el día
5	1.58-1.64	1.61	1	Fase diurna de nubes y tamaño de partículas, nieve
6	2.225-2.275	2,25	2	Propiedades diurnas de la tierra / nubes, tamaño de partículas, vegetación, nieve
7	3.80-4.00	3.9	2	Superficie y nube, niebla en la noche, fuego, vientos.
8	5.77-6.6	6.19	2	Vapor de agua atmosférica de alto nivel, vientos, precipitaciones.
9	6.75-7.15	6.95	2	Vapor de agua, vientos, precipitaciones atmosféricas de nivel medio.
10	7.24-7.44	7.34	2	Vapor de agua de nivel inferior, vientos y SO <sub>2</sub>
11	8.3-8.7	8.5	2	Agua total para estabilidad, fase nubosa, polvo, SO <sub>2</sub> , lluvia.
12	9.42-9.8	9.61	2	Ozono total, turbulencia, vientos.
13	10.1-10.6	10.35	2	Superficie y nube
14	10.8-11.6	11.2	2	Imágenes, SST, nubes, lluvia.
15	11.8-12.8	12.3	2	Agua total, ceniza, SST
16	13.0-13.6	13.3	2	Temperatura del aire, alturas y cantidades de nubes.

El sensor ABI escanea a la Tierra en tres regiones, el tiempo de escaneo de estas tres regiones depende del modo de escaneo en el que se encuentre el satélite, ver Tabla 2.7.

**Tabla 2.7:** Regiones de escaneo del sensor ABI(región utilizada en color gris), fuente de NOAA.

Región	Modo 3 Modo Flexible	Modo 4 Modo continuo	Modo 6
FullDisk	15 min +-30 seg	5 min +-5 seg	10 min +-30 seg
CONUS	5 min +-30 seg	N/A	5 min +-30 seg
Mesoescala	30 seg +-5 seg	N/A	30 seg +-5 seg

La región de Mesoescala es dinámica y se prioriza en eventos extraordinarios, ver Figura 2.6



**Figura 2.6:** Regiones de escaneo del satélite GOES-16 en proyección geoestacionaria, elaboración propia con datos de GOES-16.

El segmento terrestre consta de elementos que se encargan de gestión de la misión, generación del producto, distribución del producto, gestión empresarial e infraestructura.

El satélite GOES-16 en comparación con su serie anterior, tiene tres veces más resolución espectral, cuatro veces más resolución espacial y cinco veces más resolución temporal. Esto ayuda en gran medida a mejorar pronósticos y el monitoreo meteorológico, enviando alertas tempranas gracias a su resolución temporal, además de integrar datos de rayos, monitoreo solar y de la magnetosfera.[GOES-16,2018]

## 2.3. Satélites de Órbita Polar

### 2.3.1. Suomi-NPP

El satélite Suomi-NPP sería el primer satélite del programa NPOESS, que sería el programa de la nueva generación satelital de monitoreo meteorológico y terrestre de EUA sustituyendo a los satélites NOAA, Aqua y Terra. Sin embargo el programa se canceló por retrasos de construcción.

Ante esto, el satélite Suomi-NPP fue lanzado el 28 de Octubre del 2011 para servir como brecha entre el programa de satélites POES (NOAA) y el programa JPSS<sup>20</sup>, cuyo primer satélite es el JPSS-1 o NOAA-20, que fue lanzado el 18 de Noviembre del 2017.

El programa fue desarrollado por la NASA y es administrado por la NOAA y el DOD<sup>21</sup>.

<sup>20</sup>JPSS (Joint Polar Satellite System) es el sistema de satélites ambientales operacionales de nueva generación de órbita polar de la NOAA.

<sup>21</sup>El Departamento de Defensa de Estados Unidos (DOD) es el encargado de coordinar y supervisar todas las agencias y funciones del gobierno relacionadas directamente con la seguridad nacional.

El satélite Suomi-NPP sigue siendo operativo junto con el JPSS-1 en 2019, ya que los dos comparten los mismos instrumentos. La serie de JPSS será operativa hasta 2036 [Suomi\_NPP,2018], ver Figura 2.7.

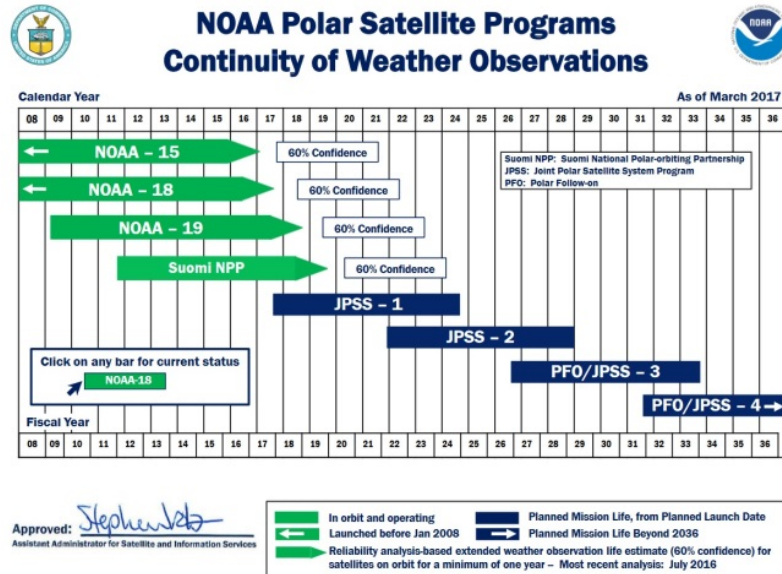


Figura 2.7: Calendario oficial de próximos satélites del programa de satélites meteorológicos de órbita polar de la NOAA aprobado hasta 2036, tomado de [https://www.satelliteconferences.noaa.gov/2017/doc/presentation/monday/NSC2017\\_Session\\_5.0\\_St.Germain.pdf](https://www.satelliteconferences.noaa.gov/2017/doc/presentation/monday/NSC2017_Session_5.0_St.Germain.pdf)

El satélite está en órbita polar heliosincronica y está a una altitud promedio de 824 km, tiene una inclinación de  $98.74^\circ$  y un periodo de 101 minutos, el cual le permite recorrer la órbita 14 veces en al día, observando casi en su totalidad a la Tierra en 24 hrs, ver Figura 2.8. La hora local en el nodo descendente<sup>22</sup> es a las 10:30 horas. [Suomi-npp-eoPortal,n.f]

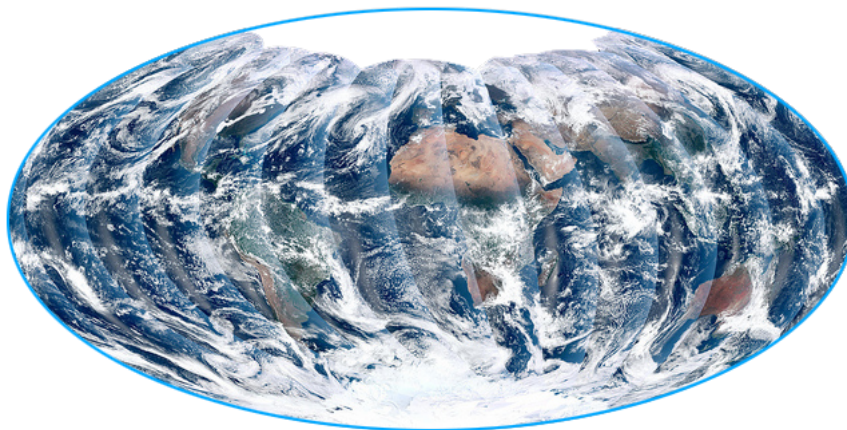


Figura 2.8: Regiones de escaneo del satélite Suomi-NPP en proyección sinusoidal, modificado de <https://earthobservatory.nasa.gov/images/76674/first-global-image-from-viirs>

Cuenta con 5 instrumentos que sirven para el monitoreo meteorológico y terrestre, ver Tabla 2.8.

<sup>22</sup>Punto donde el satélite cruza el plano de referencia moviéndose desde el hemisferio norte al hemisferio sur.



**Tabla 2.8:** Instrumentos del satélite Suomi-NPP (instrumento utilizado en color gris), fuente de NOAA.

Instrumentos	Nombre	Descripción
VIIRS	Visible Infrared Imaging Radiometer Suite	Es un radiómetro de exploración de 22 bandas que recoge la información radiométrica de la superficie terrestre, la atmósfera, la criosfera, y océanos en el visible e IR.
ATMS	Advanced Technology Microwave Sounder	Es un radiómetro microondas que ayudará a crear modelos de temperatura globales.
CrIS	Cross-track Infrared Sounder	Es un interferómetro que monitorea la presión.
OMPS	Ozone Mapping and Profiler Suite	Es un grupo de espectrómetros que mide el nivel de ozono, especialmente cerca a los polos.
CERES	Clouds and the Earth's Radiant Energy System	Es un radiómetro que detecta radiación termal, incluyendo la radiación solar.

En este trabajo se utilizaron exclusivamente datos del sensor VIIRS, ya que los productos utilizados se derivan de sus bandas. Las bandas del sensor VIIRS tienen distintas aplicaciones, así como diferentes resoluciones espaciales, ver Tabla 2.9.

**Tabla 2.9:** Características de las bandas del sensor VIIRS (bandas utilizadas en color gris), fuente de NOAA.

Canal	Longitud de onda ( $\mu\text{m}$ )	Longitud de onda central ( $\mu\text{m}$ )	Resolución espacial (m)	Objetivos de medición y uso de datos
DNB	0.5 - 0.9	0.7	750	Luces de noches.
M1	0.402-0.422	0.412	750	Color del océano, aerosoles, materia en suspensión, flujo de calor y carga en masa.
M2	0.436-0.454	0.445	750	Color del océano, aerosoles, materia en suspensión, flujo de calor y carga en masa.
M3	0.478-0.498	0.488	750	Color del océano, aerosoles, tipo de superficie, materia suspendida, flujo de calor y carga en masa.
M4	0.545-0.565	0.555	750	Color del océano, aerosoles, tipo de superficie, materia suspendida, flujo de calor y carga en masa.
I1	0.600-0.680	0.64	375 m	Imágenes, NDVI, máscara de nube, propiedades ópticas de la nube, tipo de superficie, albedo, nieve / hielo y humedad del suelo.
M5	0.662-0.682	0.672	750 m	Color del océano, aerosoles, materia suspendida, flujo de calor neto, transporte litoral y carga masiva.
M6	0.739-0.754	0.746	750 m	Corrección atmosférica, Color del océano, carga masiva.
I2	0.846-0.885	0.865	375 m	NDVI, nieve / hielo, tipo de superficie y albedo.
M7	0.846-0.885	0.865	750	Color del océano, máscara de nubes, aerosoles, humedad del suelo, flujo de calor neto y carga masiva.
M8	1.230-1.25	1.24	750	Propiedades ópticas de la nube (esenciales sobre nieve / hielo) e incendios activos.
M9	1.371-1.386	1.378	750	Máscara / cubierta de nubes (detección de cirros delgados), aerosoles, flujo de calor neto
I3	1.580-1.640	1.61	375	Imágenes de nieve / hielo (diferenciación de nubes / nieve), tipo de superficie y albedo.
M10	1.580-1.640	1.61	750	Aerosoles, propiedades ópticas de la nube, máscara / cubierta de la nube (detección de nubes / nieve), incendios activos, humedad del suelo y flujo de calor neto.
M11	2.225-2.275	2.25	750	Aerosoles (espesor óptico óptimo de aerosol sobre la Tierra), propiedades ópticas de la nube, tipo de superficie, incendios activos y flujo de calor neto.
I4	3.550-3.930	3.74	375	Imágenes (identificación de estratos bajos y oscuros) e incendios activos.
M12	3.660-3.840	3.7	750	SST (Temperatura de la superficie del mar), máscara de nube, tipo de superficie, temperatura de la superficie del hielo / tierra y aerosoles.
M13	3.973-4.128	4.05	750	SST (en zonas tropicales y durante el día), temperatura de la superficie terrestre, incendios activos y agua precipitable.
M14	8.400-8.700	8.55	750	Máscara de nube (fundamental para la detección de la fase de la nube durante la noche y propiedades ópticas de la nube
M15	10.263-11.263	10.763	750	SST, temperatura de la superficie del hielo / tierra y tipo de superficie.
I5	10.500-12.400	11.45	375	Imágenes (banda de imágenes nocturnas).
M16	11.538-12.488	12.013	750	SST, máscara de nubes, temperatura de la superficie del hielo / tierra y tipo de superficie.

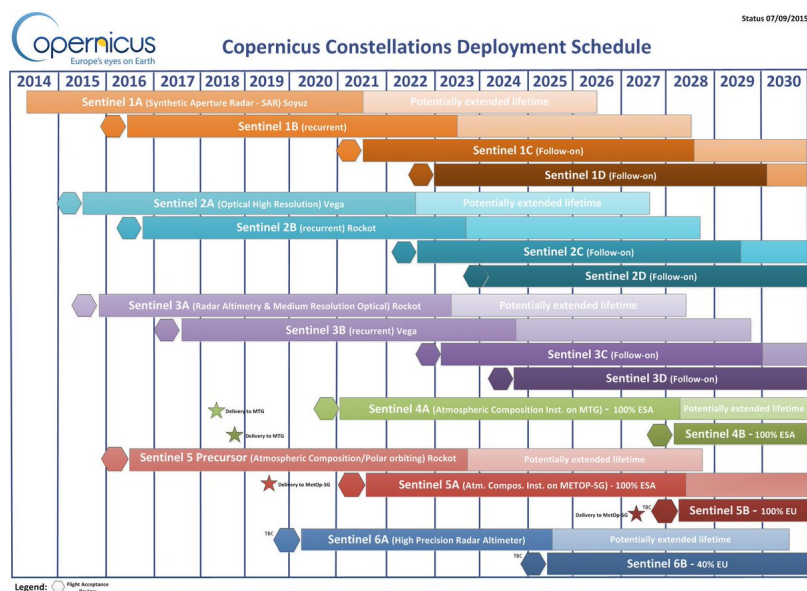
Cuenta con un segmento en tierra que se encargará de la gestión, coordinación de la misión, también distribuye y procesa los datos.

El satélite Suomi-NPP puede considerarse el primero de la serie JPSS. El manejo de los datos será muy similar al del JPSS-1, cumpliendo con dos objetivos, El primero la observación global de la atmósfera, del suelo y los océanos, y el segundo la monitorización del clima. Sus observaciones son utilizadas para crear pronósticos y brindar datos de entrada a modelos de predicción numérica del tiempo. También sirven para dar continuidad a los registros climáticos y ambientales.

### 2.3.2. Sentinel-2

El satélite Sentinel-2 pertenece al programa Copérnico, desarrollado en conjunto por la ESA y la CE<sup>23</sup>, cuyo objetivo es proveer información fiable y continua para la gestión y conservación del medio ambiente, así como combatir al cambio climático.

El programa Sentinel perteneciente al programa Copérnico<sup>24</sup>, está compuesto de 6 satélites de diferentes tipos, como satélites de radar y satélites de imágenes espectrales. El conjunto satelital busca tener datos terrestres, oceánicos y atmosféricos, teniendo diferentes propósitos para cada uno, como la medición topográfica, atmosférica, meteorológica, oceanográfica. Se planea estén en operación hasta 2031 [Sentinel-2-Portal,n.f], ver Figura 2.9.



**Figura 2.9:** Calendario oficial de próximos satélites Sentinel del programa Copérnico de la ESA y la CE aprobado hasta 2031, tomado de [http://workshop.copernicus.eu/sites/default/files/content/attachments/1.a.lukaszczyk\\_general\\_copernicus\\_introduction.pdf](http://workshop.copernicus.eu/sites/default/files/content/attachments/1.a.lukaszczyk_general_copernicus_introduction.pdf)

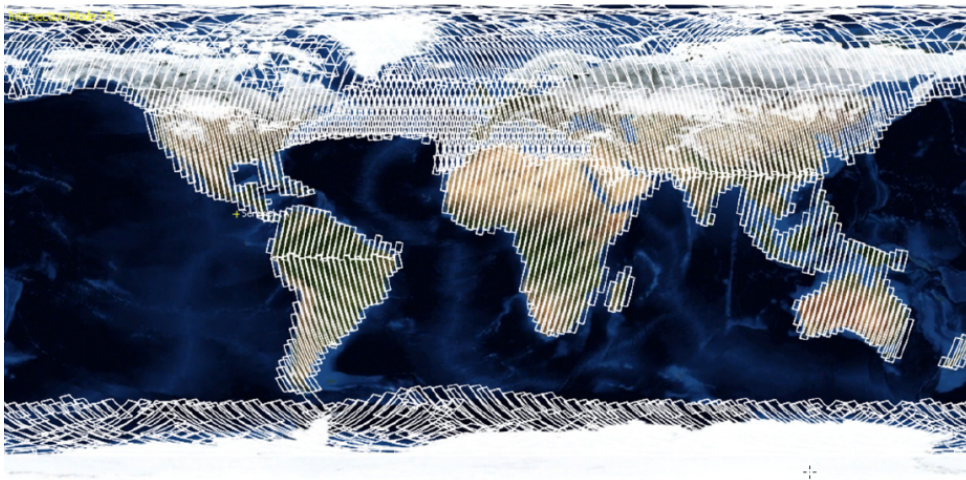
<sup>23</sup>La Comisión Europea (CE) es una de las siete instituciones de la Unión Europea, ostenta el poder ejecutivo y la iniciativa legislativa.

<sup>24</sup>Programa de la ESA y UE que pretende lograr una completa, continua y autónoma capacidad de observación terrestre de alta calidad.

Uno de los 6 satélites es Sentinel-2. Este se ha desarrollado para la adquisición de imágenes multiespectrales de alta resolución espacial y temporal, dando continuidad mejorada a las imágenes multiespectrales proporcionadas por Landsat-7 y SPOT-5.

Cuenta con un segmento espacial y uno en tierra. El segmento espacial del Sentinel-2 está dividido en dos satélites idénticos (Sentinel-2A y Sentinel-2B), que mejoran la resolución temporal de las imágenes. Sentinel-2A fue lanzado el 23 de Junio del 2015 y Sentinel-2B el 2 de Marzo del 2017. Serán sustituidos por Sentinel-2C y Sentinel-2D, en 2022 y 2023, respectivamente, alargando las operaciones del programa.[Sentinel-2- eoPortal,n.f]

Los satélites están en órbita polar heliosincronica y a una altitud promedio de 786 km, tiene una inclinación de 98.5623°. Debido a la alta resolución espacial que manejan, con franjas de barrido de 290 km de ancho, la resolución espacial es de 10 días por satélite. Con los dos satélites operativos se reduce a 5 días. Tiene un periodo de 101.65 minutos, el cual le permite cubrir casi la totalidad de la superficie terrestre en 140 vueltas aproximadamente, ver Figura 2.10. La hora local en el nodo descendente es a las 10:30 horas.[Sentinel-2- eoPortal,n.f]



**Figura 2.10:** Regiones de escaneo del satélite Sentinel-2 en proyección UTM, tomado de <https://sentinel.esa.int/web/sentinel/missions/sentinel-1/satellite-description/geographical-coverage>

El satélite Sentinel-2(2A y 2B) cuenta solo con solo un instrumento multiespectral, ver Tabla 2.10.

**Tabla 2.10:** Instrumento del satélite Sentinel-2 (instrumento utilizado en color gris), fuente de ESA.

Instrumentos	Nombre	Descripción
MSI	MultiSpectral Instrument	Es un radiómetro de empuje de 13 bandas que detecta la radiación en el visible, NIR y SWIR. Tiene un gran rendimiento de alto espectro, geométrico y espectral de las mediciones.

Las longitudes de onda de las bandas del sensor MSI varían respecto del Sentinel-2A y Sentinel-2B, ver Tabla 2.11.

**Tabla 2.11:** Características de las bandas del sensor MSI (bandas utilizadas en color gris), fuente de ESA.

Canal	Longitud de onda ( $\mu\text{m}$ )	Longitud de onda central ( $\mu\text{m}$ )	Longitud de onda ( $\mu\text{m}$ )	Longitud de onda central ( $\mu\text{m}$ )	Resolución espacial (m)	Objetivos de medición y uso de datos
	Sentinel-2A		Sentinel-2B			
1	0.5 - 0.9	0.4427	0.5 - 0.9	442.2	60	Luces de noches.
2	0.402-0.422	0.4924	0.402-0.422	492.1	10	Color del océano, aerosoles, materia en suspensión, flujo de calor y carga en masa.
3	0.436-0.454	0.5598	0.436-0.454	559	10	Color del océano, aerosoles, materia en suspensión, flujo de calor y carga en masa.
4	0.478-0.498	0.6646	0.478-0.498	664.9	10	Color del océano, aerosoles, tipo de superficie, materia suspendida, flujo de calor y carga en masa.
5	0.545-0.565	0.7041	0.545-0.565	703.8	20	Color del océano, aerosoles, tipo de superficie, materia suspendida, flujo de calor y carga en masa.
6	0.600-0.680	0.7405	0.600-0.680	739.1	20	Imágenes, NDVI, máscara de nube, propiedades ópticas de la nube, tipo de superficie, albedo, nieve / hielo y humedad del suelo.
7	0.662-0.682	0.7828	0.662-0.682	779.7	20	Color del océano, aerosoles, materia suspendida, flujo de calor neto, transporte litoral y carga masiva.
8	0.739-0.754	0.8328	0.739-0.754	832.9	10	Corrección atmosférica, Color del océano, carga masiva.
8a	0.846-0.885	0.8647	0.846-0.885	864	20	NDVI, nieve / hielo, tipo de superficie y albedo.
9	0.846-0.885	0.9451	0.846-0.885	943.2	60	Color del océano, máscara de nubes, aerosoles, humedad del suelo, flujo de calor neto y carga masiva.
10	1.230-1.25	1.3735	1.230-1.25	1376.9	60	Propiedades ópticas de la nube (esenciales sobre nieve / hielo) e incendios activos.
11	1.371-1.386	1.6137	1.371-1.386	1610.4	20	Máscara / cubierta de nubes (detección de cirros delgados), aerosoles, flujo de calor neto
12	1.580-1.640	2.2024	1.580-1.640	2185.7	20	Imágenes de nieve / hielo (diferenciación de nubes / nieve), tipo de superficie y albedo.

El segmento en tierra tiene labores de operaciones de vuelo, gestión de datos de carga útil de servicio y de procesamiento de datos.

Sentinel-2 se ha convertido en el satélite con mayor resolución espacial de libre acceso para el estudio de la cobertura terrestre y la vigilancia ambiental. Muchas aplicaciones se han derivado de su capacidad de observación, como, el seguimiento de los bosques, observación de las zonas costeras, vigilancia de las aguas continentales, vigilancia de glaciares, mapeo de la extensión del hielo, el seguimiento de la capa de nieve, mapeo y gestión de las inundaciones, entre otros. Esto ya lo hacían programas satelitales anteriores, pero ahora con mayor resolución espacial y temporal.



## Capítulo 3

# Productos meteorológicos satelitales

Los datos satelitales que fueron utilizados en los procesos, proceden de los productos satelitales. Estos productos son el resultado final de la aplicación de algoritmos previos a los valores captados por el sensor.

En este trabajo, los productos satelitales se han clasificado en dos grupos, los que están relacionados con **variables meteorológicas** y los que se desarrollan mediante **composiciones RGB**.

Los que están relacionados con **variables meteorológicas** son generados mediante algoritmos que pueden utilizar datos auxiliares o depender de otros productos, su importancia está en los valores numéricos que obtienen.

Los que se desarrollan mediante **composiciones RGB**, asignan a cada color los valores de cierta banda del sensor, con cierto o nulo tratamiento previo a los valores. Su importancia está en la asociación de elementos a los colores finales de la imagen RGB.

### 3.1. Variables meteorológicas

Los algoritmos de estos productos, siguen un diagrama de flujo para su obtención y son procesados por las administraciones de los programas satelitales, usando las bases teóricas de un algoritmo y realizando adecuaciones específicas para cada tipo de sensor. Se conocen comúnmente como datos de nivel L2.

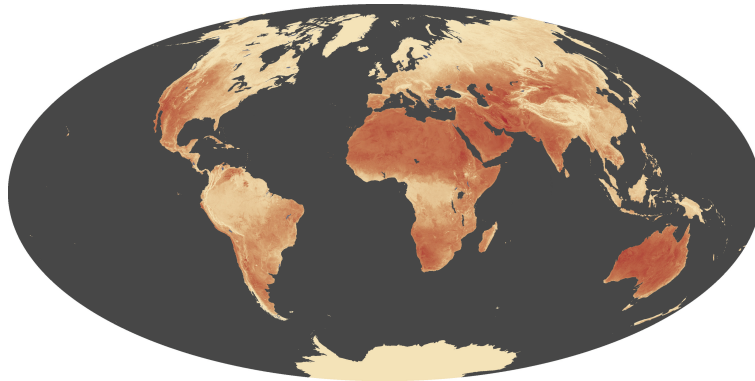
Como parte del desarrollo de los procesos automatizados, se emplearon datos de productos satelitales asociados a variables meteorológicas que fueran similares a los datos registrados en estaciones meteorológicas automáticas y boyas marinas, para posteriormente poder integrar los datos (satelital y estaciones/boyas) de forma automática en los resultados finales.

Los productos que fueron usados, son los correspondientes a los desarrollados para los sensores ABI/GOES-16 y VIIRS/Suomi-NPP, estos son:

- Land Surface Temperature.
- Sea Surface Temperature.

### 3.1.1. Land Surface Temperature

LST es un producto el cual determina la temperatura de la superficie que detecta el sensor atravesando la atmosfera. Puede ser cualquier superficie en tierra, como vegetación o pavimento, ya que el producto no se obtiene en superficies oceánicas y en cuerpos de agua, ver Figura 3.1.



**Figura 3.1:** Compuesto del producto LST de MODIS/Terra global de 2003 a 2009, modificado de <https://earthobservatory.nasa.gov/images/77779/finding-the-hottest-spots-on-earth-by-satellite>

LST es un indicador clave del presupuesto de energía de la superficie terrestre, se requiere ampliamente en aplicaciones de hidrología, meteorología y climatología. Es de fundamental importancia para la estimación de radiación en la superficie de la Tierra y para monitorear el estado de los cultivos y vegetación, así como un indicador importante tanto del efecto invernadero como del flujo de energía entre la atmósfera y el suelo.[Norman y Becker,n.f]

Puede ser la entrada a modelos climáticos y atmosféricos, así como servir de análisis climático en series de tiempo.

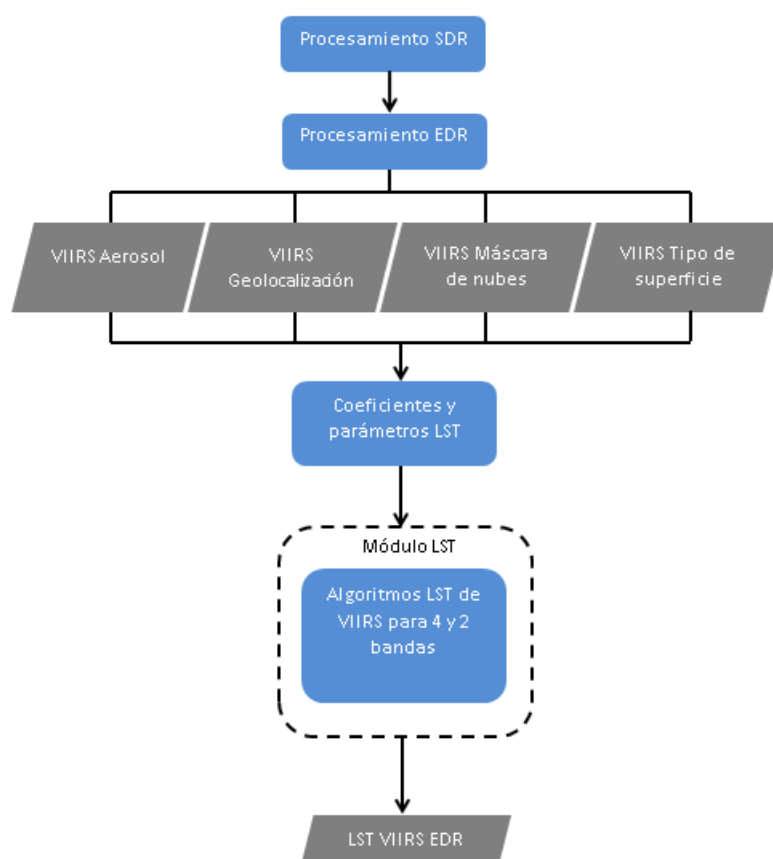
El producto LST tiene un proceso particular de obtención para el sensor VIIRS/Suomi-NP y para el sensor ABI/GOES-16.

**LST VIIRS.**

El producto LST del sensor VIIRS recupera la temperatura para cada pixel terrestre libre de nubes a una resolución de moderada de 750 m. Previamente con el uso de un módulo de procesamiento<sup>1</sup>, determina a partir de otros productos VIIRS si el algoritmo es procesado en el pixel.

En el modo operacional utiliza un algoritmo de ventana dividida<sup>2</sup> con 2 bandas del sensor, la banda 15 y 16, y cuenta con otro algoritmo opcional de ventana dividida con 4 bandas, la 12, 13, 15 y 16. LST se recupera mediante una ecuación de regresión con coeficientes para 17 diferentes tipos de superficies del IGBP<sup>3</sup>. [Godin y Vicente,2013]

Con los datos auxiliares, se calcula LST QF (Quality Flags) que sirven como banderas de decisión. Sus valores se utilizan para determinar si es posible recuperar el LST y determinar que algoritmo usar, ver Figura 3.2.



**Figura 3.2:** Diagrama de flujo del procesamiento del producto LST de VIIRS/Suomi-NPP, fuente [Godin y Vicente,2013].

<sup>1</sup>Procesamiento a lo datos de entrada y auxiliares para el cálculo del producto.

<sup>2</sup>Eliminación del efecto atmosférico a través de la diferencia de dos bandas infrarrojas térmicas (11 μm y 12 μm), la combinación no lineal de las temperaturas de brillo se aplica para la estimación de LST.

<sup>3</sup>Programa Internacional de Geosfera-Biosfera (IGBP), es un programa de investigación que estudia el fenómeno del cambio global que se desarrolló desde 1987 hasta 2015.



LST no es calculado en las siguientes condiciones.

- El píxel está nublado (LST QF en la Nube está “Nublada”)
- El píxel es un píxel del océano (QF de LandWater es ”SeaWater”).
- BT de la banda M15 está fuera del rango definido por LST.
- BT de la banda M16 está fuera del rango definido por el LST.
- El tipo de superficie de tierra está fuera del rango definido por el LST.

Los algoritmos LST de ventana dividida simple para VIIRS, son los siguientes:

**Día:**

$$LST = a_0 + a_1 T_{M15} + a_2 (T_{M15} - T_{M16}) + a_3 (\sec\theta - 1) + a_4 (T_{M15} - T_{M16})^2 \quad (3.1)$$

**Noche:**

$$LST = b_0 + b_1 T_{M15} + b_2 (T_{M15} - T_{M16}) + b_3 (\sec\theta - 1) + b_4 (T_{M15} - T_{M16})^2 \quad (3.2)$$

Donde:

$LST$  = Land Surface Temperature de VIIRS.

$a$  y  $b$  = Coeficiente de regresión LST, dependientes del tipo de superficie y condiciones de día/noche.

$\theta$  = Ángulo cenital satelital<sup>4</sup>.

$T_i$  = BT de las bandas M15 y M16 de VIIRS.

### LST GOES-16.

El producto LST del sensor ABI recupera la temperatura para cada píxel terrestre libre de nubes a una resolución de moderada de 2 km. Se calcula para los tres modos de regiones de escaneo, FullDisk, CONUS, y Mesoescala. El algoritmo está limitado en zonas de nubosidad, donde se calcula con alto o bajo grado de calidad y está caracterizado para 4 condiciones: diurna y noche, y ambientes secos y húmedos.[Yu,Xu y Chen,2010]

Se calcula mediante un algoritmo de ventana dividida, que usa la banda 14 y 15, además corrige las condiciones atmosféricas de absorción y le aplica la información de emisividad superficial prescrita<sup>5</sup>. Los coeficientes del algoritmo LST se obtuvieron a partir de un modelo de transferencia de radiación (RTM)<sup>6</sup> y un conjunto de datos

<sup>4</sup>Representa la posición del satélite relativa a la normal local.

<sup>5</sup>Proporción de radiación térmica emitida por una superficie u objeto debido a su temperatura.

<sup>6</sup>Estimación de la radiación global que llega a la superficie terrestre.

de mediciones de temperatura del suelo, están clasificados para condiciones diurnas y nocturnas, así como para condiciones secas y húmedas.[Yu,Xu y Chen,2010]

El proceso se realiza por un módulo que agrupa diversos datos de entrada, clasificándolos en tres grupos, datos del sensor ABI, productos auxiliares provenientes de ABI y productos auxiliares no provenientes de ABI, ver Figura 3.3. Estas tres entradas determinan la calidad con la que será procesado el algoritmo, y están contenidas en los valores DQF del producto final.[Yu,Xu y Chen,2010]

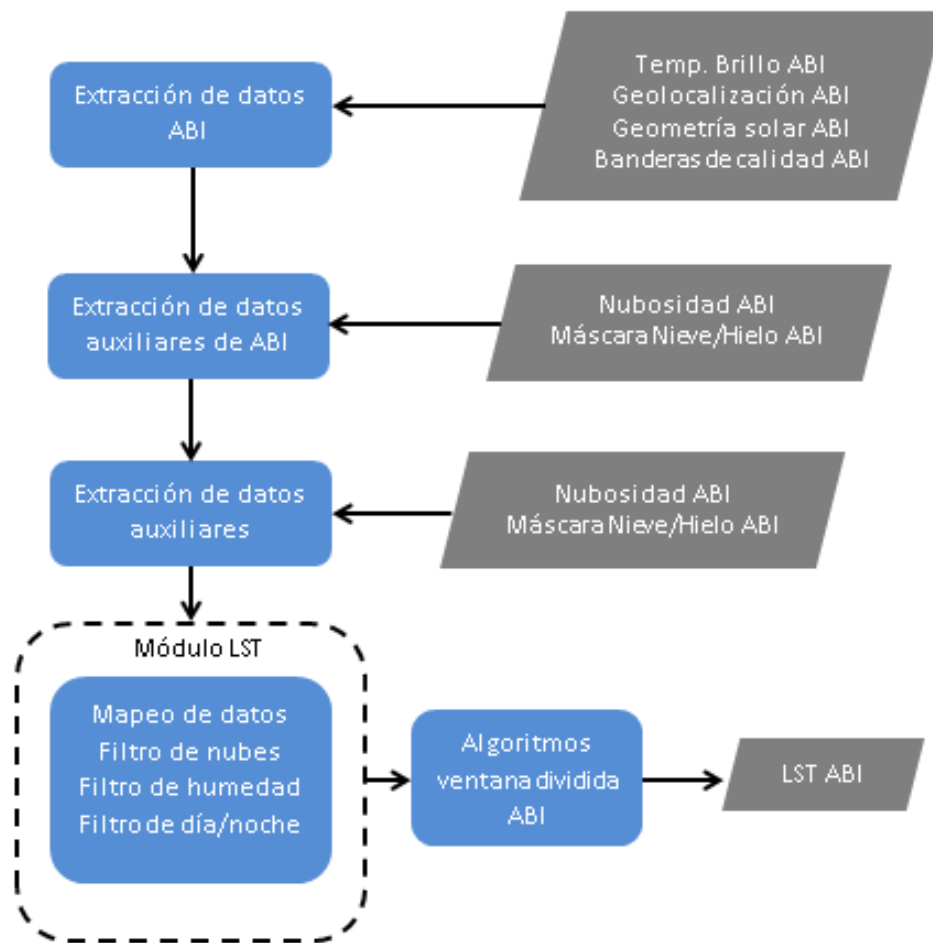


Figura 3.3: Diagrama de flujo del procesamiento del producto LST de ABI/GOES-16, fuente [Yu,Xu y Chen,2010].

Los criterios para definir los valores del DQF, son los siguientes:

- Los valores de la máscara de nubes del producto ABI nivel 2 (claro; probablemente claro; probablemente nublado y nublado).
- Valores del ángulo cenital que sirven como banderas de Día/Noche (día, ángulo cenital solar<sup>7</sup>  $\leq 85^\circ$ ; noche, ángulo cenital solar  $> 85^\circ$ ).
- Valores del tipo de superficie de los productos ABI nivel 2 (hielo de nieve, tierra y mar).
- Ángulo grande de visión (ángulo satelital cenital  $> 70^\circ$ ).
- Valores de calidad asociados a banderas de cálculo erróneo (superficies frías  $< 250\text{ K}$  &  $\geq 213\text{ K}$ ; fuera de rango de LST 213-330 K).
- Banderas de emisividad (en tiempo real de productos ABI nivel 2 y el histórico de emisividad media mensual de MODIS).

El algoritmo LST de ventana dividida para ABI es el siguiente:

$$LST = C + A_1 T_{11} + A_2 (T_{11} - T_{12}) + A_3 \varepsilon + D (T_{11} - T_{12}) (\sec \theta - 1) \quad (3.3)$$

Donde:

$LST$  = Land Surface Temperature de ABI.

$T_{11}$  y  $T_{12}$  = BT de las bandas 11 y 12 de ABI.

$\varepsilon = \frac{(\varepsilon_{11} + \varepsilon_{12})}{2}$ ; donde  $\varepsilon_{11}$  y  $\varepsilon_{12}$  son valores de emisividad espectral de la superficie terrestre en las bandas ABI 14 y 15, respectivamente.

$\theta$  = Ángulo cenital local.

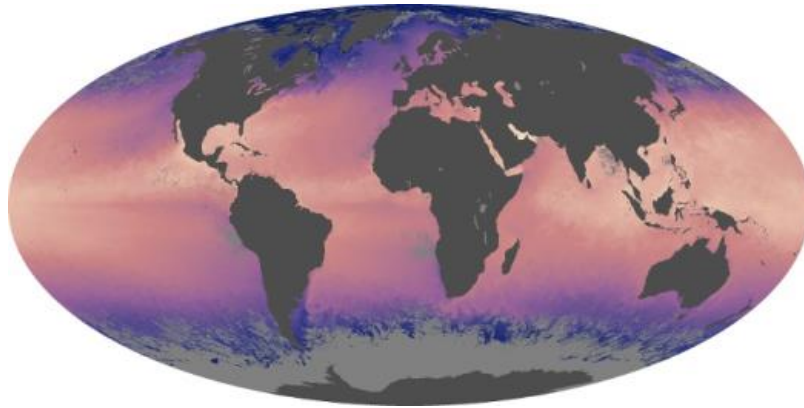
$C, A_1, A_2, A_3, D$  = Coeficientes del algoritmo.

### 3.1.2. Sea Surface Temperature

El producto SST es un indicador de la distribución de calor en la superficie del océano. Sus valores revelan las corrientes superficiales subyacentes, que son un factor determinante en el intercambio de calor y gases con la atmósfera.

<sup>7</sup>Representa la posición solar relativa a la normal local.

Es utilizado para generar modelos atmosféricos y de superficie del mar, que forman la piedra angular de los sistemas operativos de predicción oceánica. También forma parte del monitoreo de la variabilidad climática y estacional, meteorología operativa, operaciones militares y de defensa, evaluación de ecosistemas, turismo y pesquerías[Godin y Gottshall,2013], ver Figura 3.4.



**Figura 3.4:** Compuesto del producto SST de MODIS/Aqua global del mes de Julio del 2002, tomado de <https://earthobservatory.nasa.gov/global-maps/MYD28M>.

El producto SST se obtiene en superficies oceánicas y tiene un proceso particular de obtención para el sensor VIIRS/Suomi-NP y para el sensor ABI/GOES-16.

### **SST VIIRS.**

El producto SST del sensor VIIRS, recupera la temperatura para cada píxel oceánico a una resolución de moderada de 2 km. El algoritmo inicial había sido desarrollado basado en la experiencia previa con sensor MODIS y AVHRR. Después de un estudio de un conjunto de datos de emparejamientos de observaciones de SST de VIIRS con datos SST *in-situ*, se comprobó que el algoritmo SST desarrollado para los satélites EUMETSAT es el que tiene más rendimiento, el cual es una modificación a los algoritmos de ventana dividida no lineal. El algoritmo usa las bandas M15 y M16 en el día y adicionalmente la banda M12 en la noche.[Godin y Gottshall,2013]

El producto se basa en un módulo que utiliza una máscara de tierra/océano, de cobertura nubosa y otra de nieve/hielo para identificar los píxeles a procesar. Para determinar el tipo de algoritmo a usar (día/noche), el modulo usa los datos del ángulo cenital solar, y el ángulo cenital satelital del producto de geolocalización.<sup>8</sup>[Godin y Gottshall,2013]

El algoritmo usa las temperaturas de brillo calibradas de las bandas y los coeficientes para las fórmulas de ventana dividida en el día y triple ventana en la noche, el algoritmo de ventana dividida sirve como de reserva para el de la noche[Godin y Gottshall,2013], ver Figura 3.5.

<sup>8</sup>Coordenadas proyectadas del sistema para cada píxel.

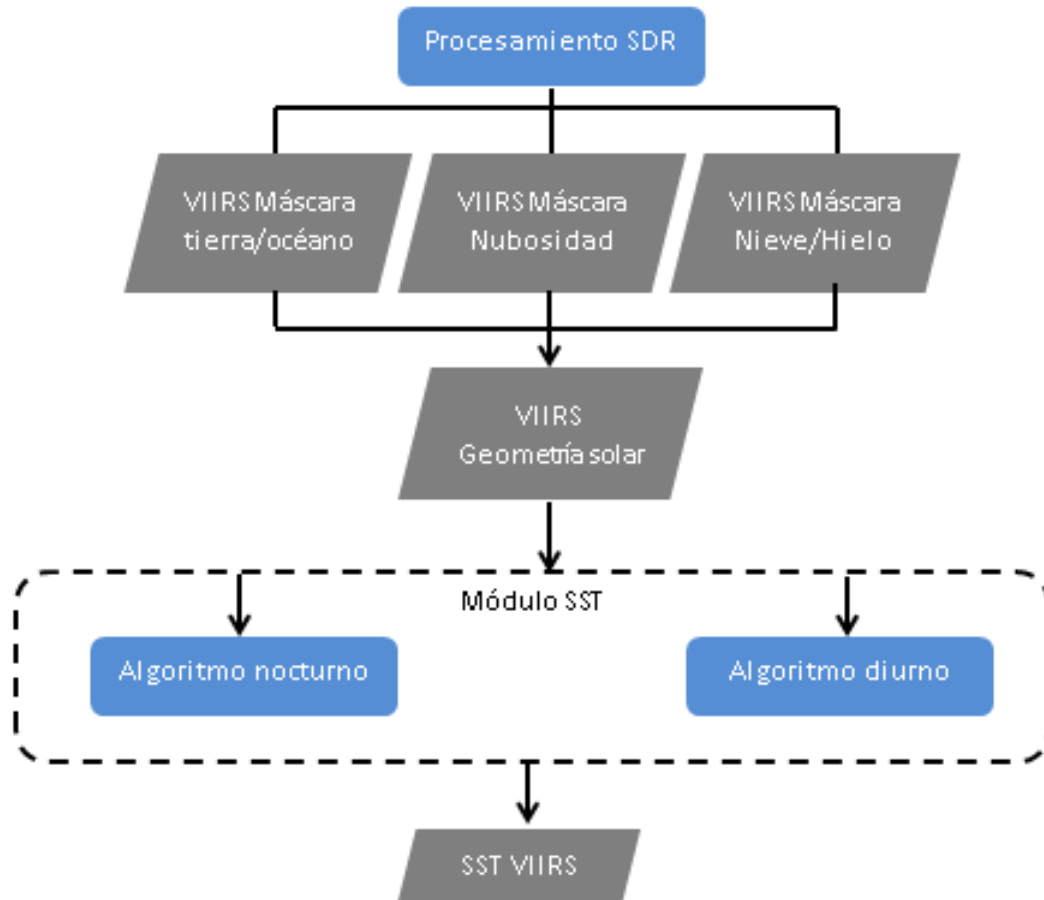


Figura 3.5: Diagrama de flujo del procesamiento del producto SST de VIIRS/Suomi-NPP, fuente [Godin y Gottshall,2013].

En condiciones de nubosidad no muy densa el algoritmo es calculado con restricciones, el cual es representado en banderas de calidad.

El algoritmo SST de día y de noche para VIIRS es el siguiente:

**Día:**

$$SST = b_0 + (b_1 + b_2 S_\theta) T_{11} + [b_3 + b_4 (T_s^0 - 273.15) + b_5 S_\theta] \Delta T_{11-12} + b_6 S_\theta \quad (3.4)$$

**Noche:**

$$SST = a_0 + (a_1 + a_2 S_\theta) T_{3.7} + (a_3 + a_4 S_\theta) \Delta T_{11-12} + a_5 S_\theta \quad (3.5)$$

Donde:

$SST$  = Sea Surface Temperature de VIIRS.

$a$  y  $b$  = Coeficiente de regresión SST, dependientes de las condiciones de día/noche.

$S_{\theta} = \sec(\theta) - 1$ , donde  $\theta$  es el ángulo cenital local.

$T_i$  = BT de las bandas en  $11 \mu m$ ,  $12 \mu m$  y  $3.7 \mu m$  de VIIRS.

$\Delta T_{11-12} = T_{11} - T_{12}$ .

$T_s^0$  = Primer valor de SST.

### SST GOES-16.

El producto SST del sensor recupera la temperatura para cada pixel oceánico a una resolución espacial moderada de 0.05 grados. Se deriva de un algoritmo probado en el sensor AHI del satélite Himawari-8. Usa perfiles atmosféricos<sup>9</sup> de vapor de agua y temperatura de un modelo de predicción numérica del tiempo, además de un modelo de transferencia radiativa, para corregir el algoritmo multispectral de los sesgos regionales y estacionales<sup>10</sup>, debido a las condiciones atmosféricas cambiantes.[PODACC,n.f]

El cálculo de SST se realiza por medio de un módulo desarrollado por el grupo AIT<sup>11</sup> de ABI. La secuencia con la que es procesado es importante, ya que el módulo SST rastrea sesgos globales entre el SST y la BT, y las promedia secuencialmente sobre varias observaciones. Estos sesgos son leídos y actualizados por el modulo al inicio del procesamiento.[Ignatov,2010]

El módulo puede ocupar las bandas 7, 11, 13, 14, 15 del sensor ABI. Utiliza además un archivo de configuración, la máscara de nubes y hielo de ABI, datos auxiliares estáticos y dinámicos, BT de pixeles sin nubosidad y su jacobiano<sup>12</sup>, simulados con el modelo de transferencia radiativa comunitaria(CRTM) en la cuadrícula GFS<sup>13</sup> bilinealmente interpolados a los píxeles del sensor. Usa también datos de Reynolds SST<sup>14</sup> igualmente interpolados bilinealmente a los píxeles del sensor[Ignatov,2010], ver Figura 3.6.

---

<sup>9</sup>Valores en diferentes capas de la atmósfera.

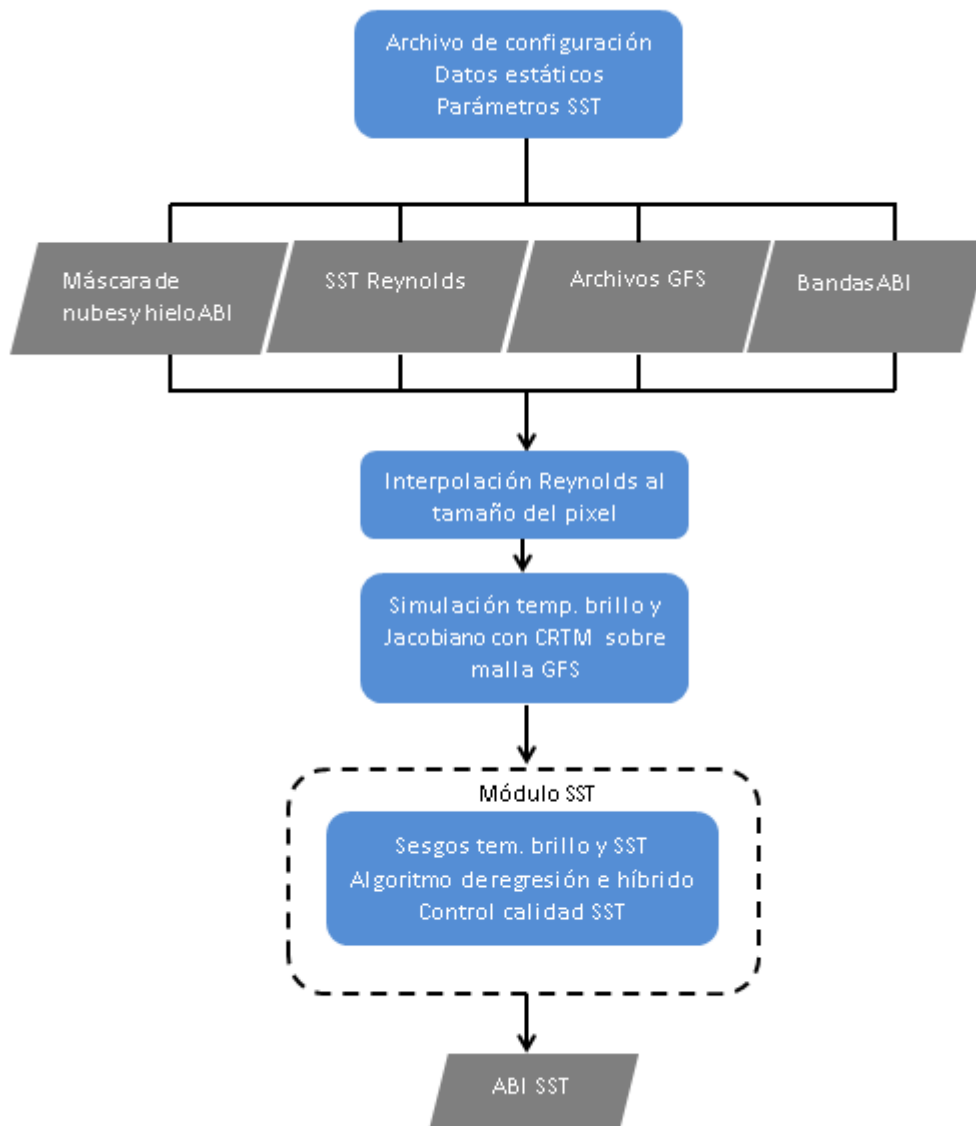
<sup>10</sup>Diferencia de los valores calculados respecto a la localización y el tiempo.

<sup>11</sup>Proporciona soporte técnico para las tareas de desarrollo e integración del algoritmo del sensor ABI.

<sup>12</sup>Derivada de una función multivariable.

<sup>13</sup>GFS (Global Forecast System), es un modelo numérico de predicción meteorológica, se actualiza cuatro veces al día con predicciones que alcanzan los 16 días.

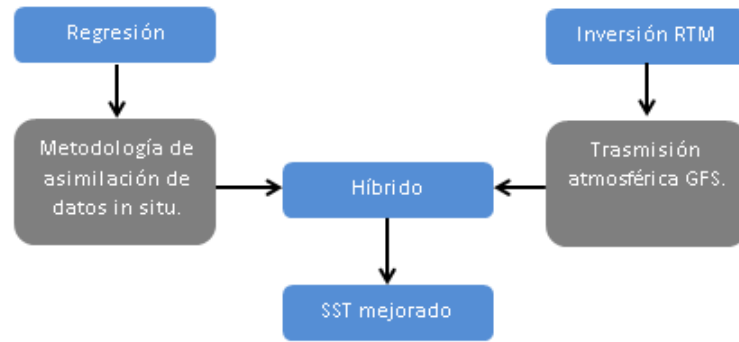
<sup>14</sup>Producto de análisis de SST de alta resolución, desarrollado con datos de AVHRR desde 1981, AMSR y datos *in situ* de boyas y barcos.



**Figura 3.6:** Diagrama de flujo del procesamiento del producto SST de ABI/GOES16, fuente [Ignatov,2010].

El algoritmo utiliza la recuperación de regresión física<sup>15</sup> híbrida para producir un producto más preciso, este algoritmo combina los puntos fuertes del algoritmo de regresión (basado en datos *in-situ*) y el de inversión (basada en datos del GFS), cuyo resultado está en los coeficientes utilizados [Ignatov,2010], ver Figura 3.7.

<sup>15</sup>Proceso estadístico para estimar las relaciones entre variables.



**Figura 3.7:** Combinación de algoritmo SST de regresión e inversión para ABI/GOES-16, fuente [Ignatov,2010].

La obtención del SST se realiza durante el día y la noche para cada píxel sin nubes, indicados en la máscara de nube ABI. Se calcula para las tres regiones de escaneo, FullDisk, CONUS, y Mesoescala.

El producto SST, cuenta con banderas de calidad QC obtenidas a partir de pruebas realizada por el modulo. El píxel puede estar clasificado como óptimo, sub-óptimo y pobre para su uso como SST.[Ignatov,2010]

El algoritmo SST de regresión e híbrido para ABI es el siguiente:

**Regresión:**

$$SST = a_0 + a_1 T_{11} + a_2 (T_{FG} - 273.15)(T_{11} - T_{12}) + a_3 (T_{11} - T_{12})(\sec\theta - 1) \quad (3.6)$$

**Híbrido:**

$$SST = b_0 + b_1 (T_{11} - T_{CS11}) + b_2 (T_{FG} - 273.15)((T_{11} - T_{CS11}) - (T_{12} - T_{CS12})) + b_3 ((T_{11} - T_{CS11}) - (T_{12} - T_{CS12}))(\sec\theta - 1) \quad (3.7)$$

Donde:

$SST$  = Sea Surface Temperature de ABI.

$a$  y  $b$  = Coeficiente de regresión SST, dependientes del tipo de algoritmo regresión/híbrido.

$\theta$  = Ángulo cenital local.

$T_i$  = BT de las bandas en  $11 \mu m$  y  $12 \mu m$  de ABI.

$T_{FG}$  = Primer valor de SST.

$T_{CSi}$  = Funciones de las bandas en  $11 \mu m$ ,  $12 \mu$ ,  $T_{FG}$ ,  $\theta$  y un vector de variables atmosférica de GFS.



### 3.2. Compuestos RGB

Los compuestos RGB son usados más de forma genérica la mayoría de las veces sin ningún tratamiento previo a los datos, generalmente provenientes de sensores multiespectrales, pues se adaptan de mejor manera.

Están basados en la posibilidad de generar color por medio de valores en un display<sup>16</sup> de computadora, el cual están compuestos por valores del 0 al 255(en un display de 8 bits) para el rojo, verde y azul. Por lo tanto, las coordenadas del rojo serán (255, 0, 0), del verde (0, 255, 0) y del azul (0, 0, 255). La ausencia de color, es decir el negro corresponde al punto (0, 0, 0). La combinación de dos colores a nivel 255 con un tercero a nivel 0 da lugar a tres colores intermedios: el amarillo (255, 255, 0), el cian (0, 255, 255) y el magenta (255, 0, 255). El blanco se forma con los tres colores primarios a su máximo nivel (255, 255, 255). La escala de grises es la diagonal que une el blanco y el negro[Bense,2007], ver Figura 3.8.

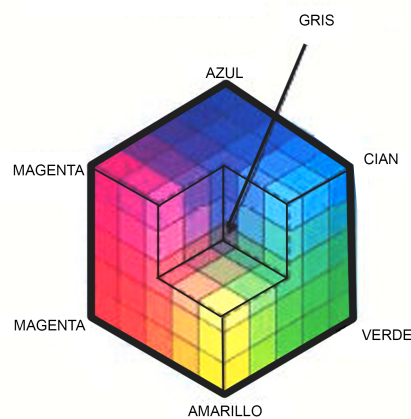


Figura 3.8: Cubo de colores RGB, modificado de <http://www.teledet.com.uy/tutorial-imagenes-satelitales/combinaciones-colores.htm>.

De esta forma los datos de las bandas de los sensores son asignados como los valores de colores primarios y presentar diversas combinaciones. Algunas combinaciones resultan ser de gran utilidad al poder resaltar características de interés en el estudio desarrollado, haciendo uso de bandas del visible e infrarrojo.

Existen diversos compuestos RGB desarrollados para diversas áreas específicas, como la ambiental, geológica, agronómica, hidrológica, meteorológica, entre otras. También pueden llegar a desarrollarse para sensores específicos con un tratamiento previo a los valores de las bandas antes de la asignación a los colores primarios. Estos se conocen más como productos RGB del sensor.

Los que se usaron en este trabajo son los desarrollados para los sensores ABI/GOES-16, VIIRS/Suomi-NPP y MSI/Sentinel-2, estos son:

<sup>16</sup>Principal dispositivo de salida (interfaz), que muestra datos o información al usuario.

- True Color.
- Fire Temperature.
- SWIR.

### 3.2.1. True Color

TC hace referencia al compuesto en el que los valores de las tres bandas correspondientes a las longitudes de onda de la luz visible, son asignadas a sus mismos colores (Rojo, Verde y Azul). Lo que genera una imagen con el color más cercano al real, parecido a observar la Tierra desde el espacio exterior.

Este compuesto tiene aplicaciones útiles en la detección de humo, neblina y polvo. También gracias a que las bandas del visible tienen buena penetración en los cuerpos de agua, es usada para observar características como turbidez, corrientes, batimetría, sedimentos, entre otros.[Bense,2007]

#### Suomi-NPP.

En el sensor VIIRS los datos de la banda M3 (Azul  $0.488 \mu\text{m}$ ), la banda M4 (Verde  $0.555 \mu\text{m}$ ) y banda M5 (Rojo  $0.672 \mu\text{m}$ ) se asignan a los valores RGB de la imagen, produciendo una imagen en TC.[Seaman,Miller y Hillger,2013]

La resolución espacial de la imagen en TC de VIIRS, le permite tener diversas aplicaciones, como diferenciar niebla de nubes, detectar tormentas de polvo, humo, fitoplancton en los océanos, sedimentos e inundaciones[Seaman,Miller y Hillger,2013] ver Figura 3.9.

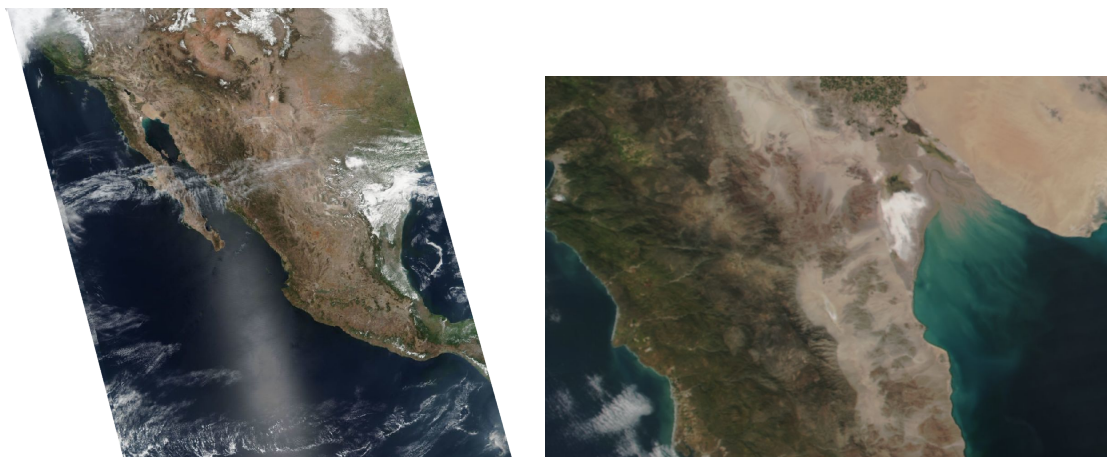
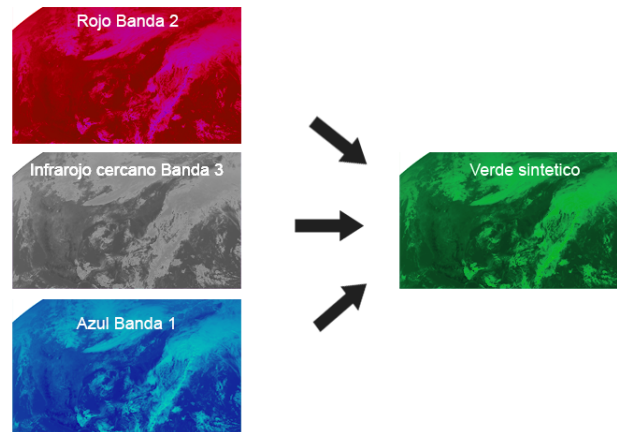


Figura 3.9: RGB TC de VIIRS/Suomi-NP, elaboración propia con datos de Suomi-NPP.

**GOES-16.**

El sensor ABI carece de una banda en la longitud del verde, por lo que se crea un verde sintético a partir de los valores de la banda 1 (Azul  $0.47 \mu\text{m}$ ), banda 2 (Rojo  $0.64 \mu\text{m}$ ) y la banda 3 (Veggie  $0.86 \mu\text{m}$ ), ver Figura 3.10. El uso de la banda 3 es importante porque imita la reflectividad presente en una banda en la longitud de onda del verde. Los coeficientes para calcular el verde, fueron obtenidos por medio de una relación lineal simple. [Lindstrom, Bah, Schmit y Kohrs, n.f ]



**Figura 3.10:** Generación de verde sintético con bandas de ABI, modificado de <https://www.slideserve.com/ayanna/history-of-true-color-satellite-imagery-true-color-imagery-from-jpss-viirs-and-goes-r-abi>.

El TC de ABI se calcula aplicando una corrección gamma<sup>17</sup> a las bandas al resultado obtenido con la siguiente fórmula:

$$V_s = (0.48358168)B_2 + (0.45706946)B_1 + (0.06038137)B_3 \quad (3.8)$$

Donde:

$V_s$  = Verde sintético.

$B_i$  = Bandas de radiancia 1,2,3 de ABI.

Si un fenómeno tiene un color particular o distintivo, tendrá una notable presencia en una imagen de TC de ABI. Ejemplos de esto incluyen cubiertas de nieve, nubosidad, polvo, humo, ceniza, sedimentos y vegetación, ver Figura 3.11. La resolución temporal de GOES-16 permite un monitoreo continuo de estos fenómenos. [Lindstrom, Bah, Schmit y Kohrs, n.f ]

Las limitaciones sobre este producto, es que solo se puede obtener en escaneos diurnos del sensor.

<sup>17</sup>Operación no lineal que se usa para codificar y decodificar luminancia de un vídeo o imagen.

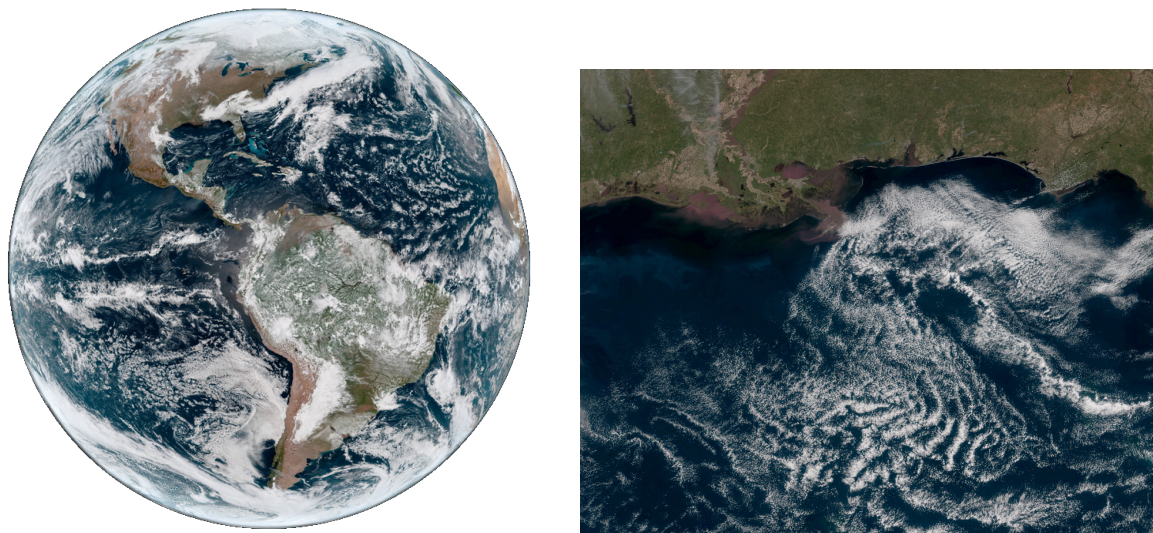


Figura 3.11: Producto RGB True color de ABI/GOES-16, elaboración propia con datos de GOES-16.

### Sentinel-2.

En el sensor MSI los datos de la banda 2 (Azul  $0.4924 \mu\text{m}$ ), la banda 3 (Verde  $0.5598 \mu\text{m}$ ) y banda 4 (Rojo  $0.6646 \mu\text{m}$ ) se asignan a los valores RGB de la imagen, produciendo una imagen en TC.

La alta resolución espacial de la imagen en TC, permite realizar monitoreo de suelo, glaciares, bosques, inundaciones, humo, polvo y vigilancia oceanográfica, ver Figura 3.12.

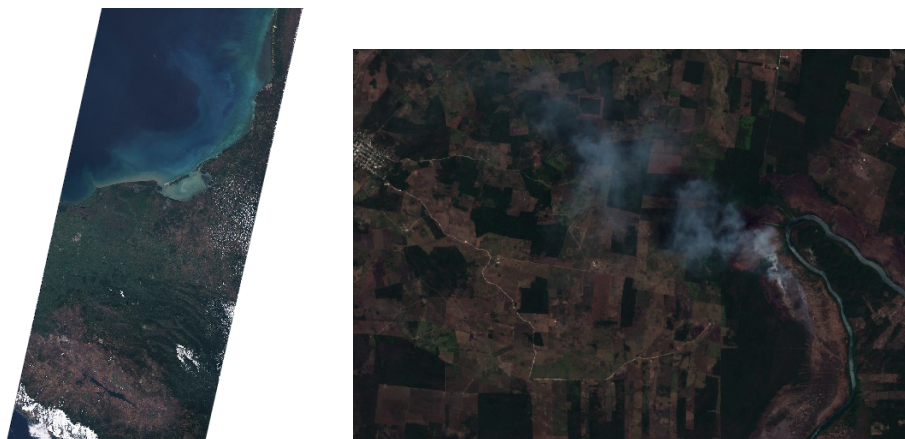


Figura 3.12: Producto RGB TC de MSI/Sentinel-2, tomada de <https://www.sentinel-hub.com/>.

### 3.2.2. Fire Temperature

El compuesto FT permite identificar incendios en diferentes intensidades, por medios de una combinación de colores para distinguirlos fácilmente. Lo hace por medio de datos en longitudes de onda que van desde el infrarrojo cercano (NIR), hasta el infrarrojo de onda corta (SWIR).

Esto porque a medida que aumenta la longitud de onda en este rango, la contribución de las fuentes de emisión de la Tierra aumenta y la contribución del sol disminuye. Como resultado, solo los puntos calientes más cálidos aparecen en las longitudes de onda cercanas a  $1.6 \mu\text{m}$ , ya que tienen que mostrar la radiación del sol que se refleja en la superficie de la Tierra. En longitudes de onda cercanas a  $3.7 \mu\text{m}$ , los puntos calientes de los incendios producen más radiación que la cantidad solar reflejada. La longitud de onda cercana a  $2.2 \mu\text{m}$  está en el medio, y solo puede mostrar los incendios relativamente fríos o pequeños.[Seaman,2013]

El producto RGB permite realizar monitoreo de incendios, con una interpretación más sencilla de las imágenes.

Combinado con otros datos como los de FIRMS<sup>18</sup>, puede servir como herramienta para la detección temprana de incendios.

#### GOES-16.

El producto RGB de FT de ABI se calcula con los valores de la banda 7 ( $3.9 \mu\text{m}$ ), banda 6 ( $2.25 \mu\text{m}$ ) y banda 5 ( $1.6 \mu\text{m}$ ), en BT. Estos son asignados a los valores RGB de la imagen respectivamente.

En la banda 7, la saturación de la temperatura brillo del píxel es alrededor de 500 K, que corresponde a fuegos de intensidad baja, por lo tanto los puntos calientes se ven de color rojo. Los incendios de alta intensidad rondan los 1400 K, la cual es cercana a la saturación de BT de la banda 5. Por lo tanto los incendios en el producto RGB pasaran de rojo, pasando por amarillo a blanco en grado de intensidad.[RAMMB,n.f]

El producto está limitado para zonas sin nubosidad y puede mostrar falsos incendios, como en zonas superficiales de alta emisividad, como regiones áridas, ver Figura 3.13.

---

<sup>18</sup>FIRMS (Fire Information for Resource Management System) distribuye datos de incendios activos en tiempo casi real dentro de las 3 horas posteriores al paso superior de MODIS y VIIRS. <https://earthdata.nasa.gov/earth-observation-data/near-real-time/firms>

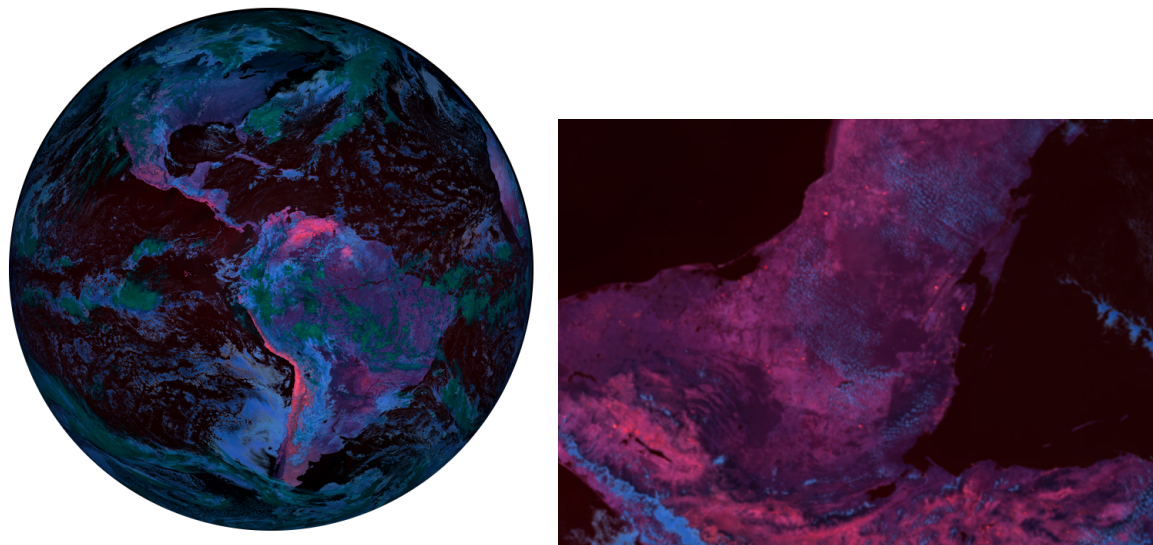


Figura 3.13: Producto RGB FT de ABI/GOES-16, elaboración propia con datos de GOES-16.

### Suomi-NPP.

El producto RGB de FT de VIIRS, se calcula con los valores de la banda M-12 ( $3.7 \mu\text{m}$ ), banda M-11 ( $2.25 \mu\text{m}$ ) y banda M-10 ( $1.61 \mu\text{m}$ ), en BT. Estos son asignados a los valores Rojo, Verde y Azul de la imagen respectivamente. [Seaman, 2013]

Al tener casi la misma asignación de longitudes de onda de las bandas a los valores RGB que en el sensor ABI, la interpretación de los colores es similar. También las limitaciones son similares a las de ABI incluyendo además que la resolución temporal es más baja, pero como atributo, su resolución espacial le permite detectar incendios en áreas más pequeñas, ver Figura 3.14.

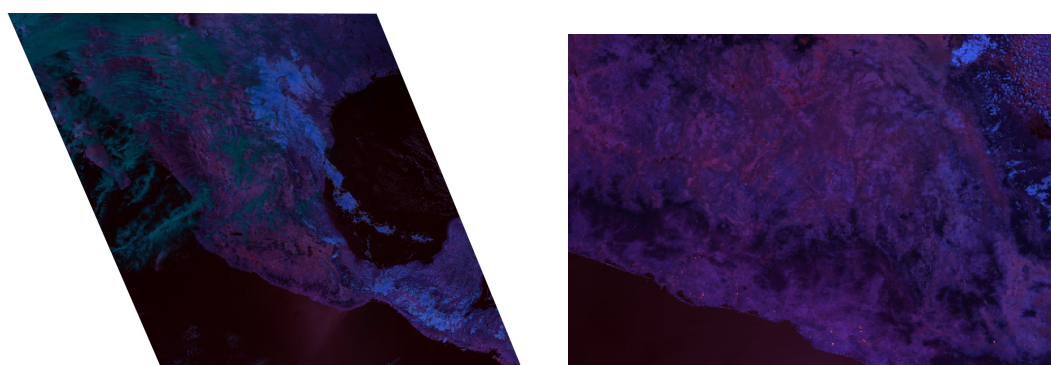


Figura 3.14: Producto RGB FT de VIIRS/Suomi-NPP, elaboración propia con datos de Suomi-NPP.

### SWIR.

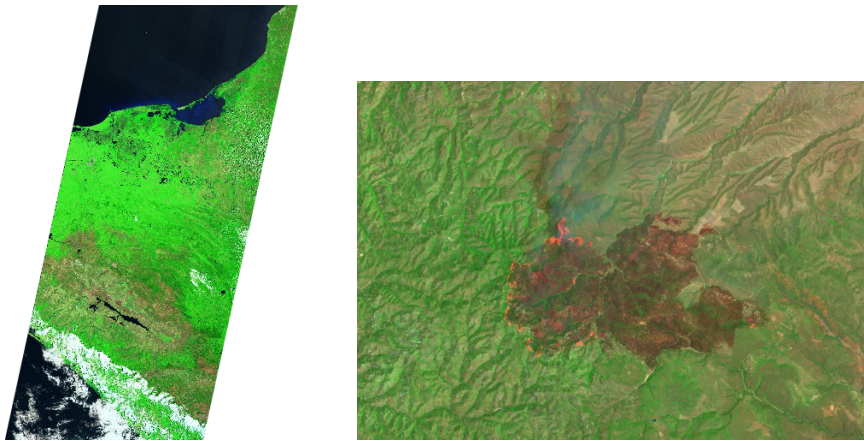
El compuesto SWIR o infrarrojo de onda corta combina datos del infrarrojo de onda corta (SWIR), infrarrojo cercano (NIR) y luz verde, asignándolos a los valores RGB de la imagen respectivamente.

Este compuesto proporciona una vista más clara de los incendios activos y las áreas quemadas. Los incendios activos tienen una coloración roja brillante, mientras que las áreas con tierra consumida por fuego pueden verse claramente con tonos rojos más oscuros. Las áreas de vegetación y urbanas son verdes.

El compuesto permite identificar claramente el comportamiento del incendio. Esto es posible solo en satélites de alta resolución, ya que con una resolución más baja, la distinción entre el área activa y el área quemada del incendio se hace muy complicada. Es usado más en zonas donde se tiene conocimiento de un incendio activo y no para su detección.

### **Sentinel-2.**

En el sensor MSI los datos de la banda 12( $2.2024 \mu\text{m}$ ), banda 8( $0.8328 \mu\text{m}$ ) y la banda 3( $0.5598 \mu\text{m}$ ), son asignados a los valores RGB de la imagen respectivamente, produciendo una imagen en SWIR, ver Figura 3.15.



**Figura 3.15:** Producto RGB SWIR de MSI/Sentinel-2, tomada de <https://www.sentinel-hub.com/>.

## Capítulo 4

# Metodología

La metodología descrita hace referencia al proceso de datos, de los tres satelitales de manera automática, mediante módulos para el manejo de datos espaciales disponibles en el lenguaje de programación Python, ver Figura 4.1

Los datos satelitales corresponden a formatos asociados a diferentes niveles de adquisición, los cuales fueron procesados de forma automática, bajo el siguiente esquema general:

- Extracción.
- Tratamiento.
- Georreferencia.
- Reproyección.
- Mapeo.

Los pasos descritos son utilizados como la base metodológica para la elaboración de los procesos, los cuales tienen un desarrollo mayor y emplean módulos adicionales en cada uno.

Cada etapa es diferente en cada formato, ya que aunque los datos procedan de diferentes tipos de satélite, el formato no está relacionado con estos y cualquier formato puede presentarse como forma de adquisición.

El formato de los datos es el principal factor que determina la forma de procesamiento y los módulos ocupados. En la Tabla 4.1 se muestra el tipo de dato, el nivel del producto y el servidor de donde fueron obtenidos los datos de cada satélite.



Tabla 4.1: Características de los datos satelitales utilizados.

Satélite.	Formato.	Nivel de Producto.	Producto.	Servidor.
GOES-16	NetCDF4	L1 y L2	radiancias, Reflectancias y BT, LST	LANOT-UNAM
Suomi-NPP	GeoTIFF	L1 y L2	radiancias y BT, SST	LANOT-UNAM
Sentinel-2	WMS	L2	Compuesto RGB	Sentinel-Hub

La Figura 4.1 muestra que nuevos módulos se fueron requiriendo en cada etapa de la metodología para cada formato.

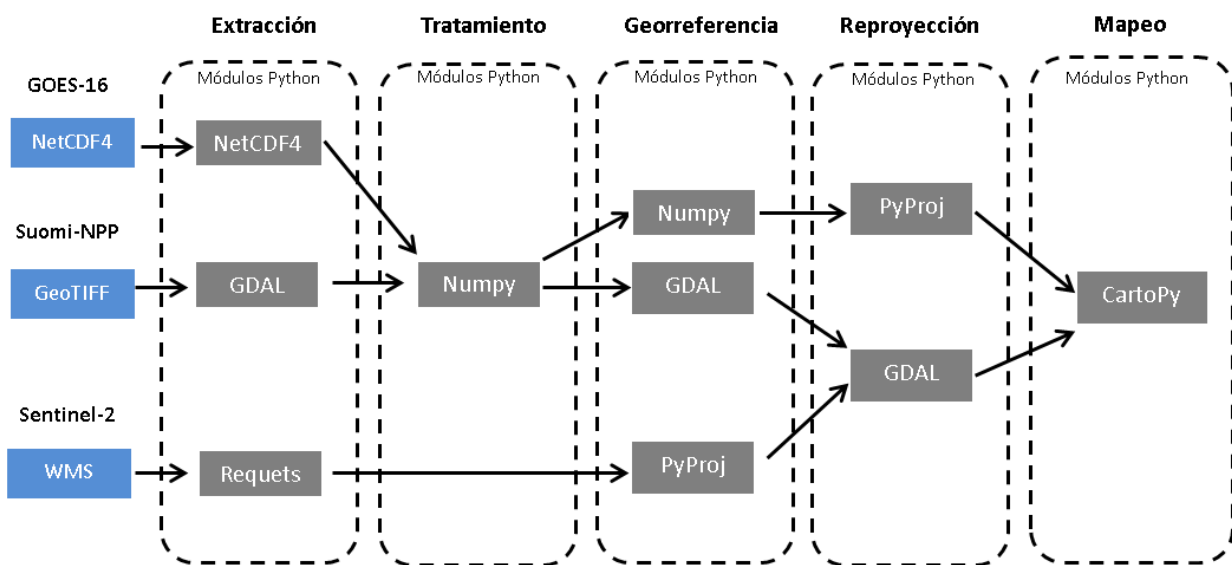


Figura 4.1: Distribución del uso de módulos de Python asociados al formato, usados en la metodología.

## 4.1. Extracción

### Datos GOES-16.

Los datos utilizados de nivel L1b y L2 del satélite GOES-16 fueron adquiridos en formato NetCDF4 a través del servidor de GOES del LANOT, el cual obtiene los datos de nivel L1b a través de una antena, la cual procesa la señal del satélite y los transforma con el sistema GRB<sup>1</sup> a archivos NetCDF4. Los archivos de nivel L2 se procesan a través del software CSPPGeo<sup>2</sup>, que ocupa los archivos L1b como datos de entrada, generando archivos L2 en formato

<sup>1</sup>GRB (GOES Rebroadcast), proporciona el relé primario del relé espacial de difusión directa, en tiempo real, calibrado y de resolución completa de los datos L1b para ABI y L2 para GML.

<sup>2</sup>CSPP Geo genera productos geofísicos a partir de datos satelitales geoestacionarios. <http://cimss.ssec.wisc.edu/csppgeo/>

NetCDF4.

Las características del formato NetCDF4 están descritas en el **Anexo de Formatos**.

Los archivos NetCDF4 de nivel L1 y L2 de GOES-16, tienen un formato de nombre genérico, el cual ayuda a mantener una organización en un conjunto de datos y permite conocer información importante del archivo. El formato es el siguiente:

```
OR_ABI-L2-CMIPF-M3C09_G16_sYYYYJJHHMMSSs_eYYYYJJHHMMSSs_cYYYYJJHHMMSSs.nc
```

En la Tabla 4.2 se describen los elementos del formato del nombre.

**Tabla 4.2:** Elementos del formato en el nombre estándar de los datos NetCDF4 de GOES-16, fuente de NOAA.

Clave	Ejemplo	Descripción
OR	Operational System Real-Time Data	Estado del satélite.
ABI-L2	Advanced Baseline Imager Level 2+	Sensor y nivel de procesamiento.
CMIPF	Cloud and Moisture Image Product – Full Disk	Producto y región de escaneo.
M3 / M4/ M6	ABI Mode 3, ABI Mode 4 y ABI Mode 6	Modo de escaneo.
C09	Channel 9	Numero de banda.
G16	GOES-16	Satélite.
sYYYYJJHHMMSSs	Año-DíaJuliano-Hora-Minuto-Segundo	Tiempo de inicio de escaneo.
eYYYYJJHHMMSSs	Año-DíaJuliano-Hora-Minuto-Segundo	Tiempo de finalización de escaneo.
cYYYYJJHHMMSSs	Año-DíaJuliano-Hora-Minuto-Segundo	Tiempo de creación del archivo.

El módulo de Python necesario para la extracción de datos, es el módulo *NetCDF4*. Este módulo contiene los manejadores para la escritura y lectura de archivos NetCDF4 con Python.

La extracción de datos, es realizada por medio de asignaciones en modo lectura de un archivo NetCDF4 a una variable, esto por medio de la función constructor *Dataset*<sup>3</sup> del módulo *NetCDF4*.

```
>>> from netCDF4 import Dataset
>>> path = 'OR_ABI-L1b-RadF-M3C01_G16_s20183640002188_e20183640004561_c20183640005004.nc'
>>> nc = Dataset(path, 'r')
```

El nombre del archivo NetCDF4, es un parámetro de la función constructor *Dataset*, el cual debe ser un tipo de dato string<sup>4</sup>. El otro parámetro es el modo de apertura definido por un carácter.

El NetCDF4 es abierto en modo lectura 'r', si se omite este parámetro, el valor por defecto es el del modo de lectura 'r'. Los caracteres de los demás métodos de apertura, son similares a los usados en manejo de archivos de Python.

<sup>3</sup>Termino usado para referirse a un conjunto de datos, habitualmente tabulados.

<sup>4</sup>Tipo de datos utilizado en programación usado para representar texto en lugar de números.

La variable creada contiene toda la información del NetCDF4 y es un objeto de la clase *Dataset*, por lo cual puede llamar a las propiedades y métodos del objeto.

El llamado de la variable, devuelve un metadato general, en donde se muestran todas las propiedades que componen al NetCDF4 y su valor. Sirve como un primer acercamiento a la exploración del archivo.

```
>>> print(nc)
type 'netCDF4._netCDF4.Dataset'
root group (NETCDF4 data model, file format HDF5):
  naming_authority: gov.nesdis.noaa
  Conventions: CF-1.7
  Metadata_Conventions: Unidata Dataset Discovery v1.0
  standard_name_vocabulary: CF Standard Name Table (v35, 20 July 2016)
  institution: DOC/NOAA/NESDIS > U.S. Department of Commerce, National Oceanic and
  Atmospheric Administration, National Environmental Satellite, Data, and Information
  Services
  project: GOES
  .
  .
  .
```

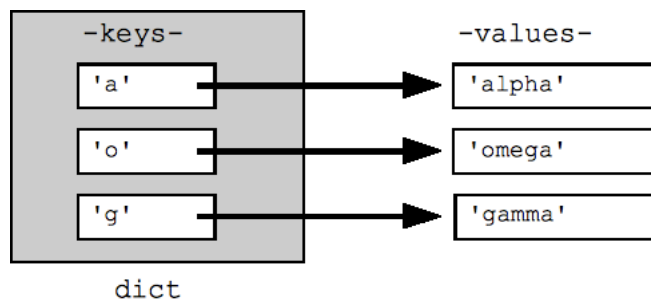
Se puede tener acceso solo a ciertas propiedades del objeto *Dataset*, mediante la colocación de un punto enseguida del nombre de la variable *nc*.

```
>>> nc.project
u'GOES'
```

Las dimensiones y las variables, son los dos principales elementos en la estructura del NetCDF4, forman parte de las propiedades del objeto, por lo que son llamadas con un punto después del nombre de la variable. Son datos de tipo diccionario.

```
>>> type(nc.dimensions)
<class 'collections.OrderedDict'>
>>> type(nc.variables)
<class 'collections.OrderedDict'>
```

Los diccionarios son un tipo de dato nativo de Python, almacenan un conjunto no ordenado de valores asociados a una llave, ver Figura 4.2. La llave puede ser cualquier otro tipo de dato, incluso otro diccionario.



**Figura 4.2:** Estructura de datos en los diccionarios de Python, tomada de <https://developers.google.com/edu/python/dict-files>.

Para conocer las llaves del diccionario de dimensiones y variables, se usa la función *keys*.

```
>>> nc.dimensions.keys()
[u'y', u'x', u'number_of_time_bounds', u'band', . . .]
>>> nc.variables.keys()
[u'Rad', u'DQF', u't', u'y', u'x', u'time_bounds', . . .]
```

El método devuelve las llaves en un formato de dato de tipo string, estas son necesarias para la extracción de los datos del diccionario.

Los datos que contienen el diccionario de dimensiones pertenecen a la subclase *dimensions*, el cual es llamado con su respectiva llave. Muestra información de la dimensión, como el nombre y el tamaño de la dimensión.

```
>>> nc.dimensions['x']
<type 'netCDF4._netCDF4.Dimension': name = 'x', size = 5000
```

Los valores pueden ser llamados mediante la colocación de un punto y el nombre seguido del objeto, ya que estos son parámetros del objeto.

```
>>> nc.dimensions['x'].name
u'x'
```

En el caso del diccionario de variables, los objetos contienen más información, esta puede clasificarse en dos grupos:

- Metadato de la variable.
- Datos de la variable.

Para conocer el **metadato de la variable**, se coloca la llave en un primer corchete, después de colocar la llave del diccionario de variables.

El metadato, es un objeto de la subclase *Variable*, el cual contiene información propia de la variable, como la descripción general, el nombre estándar, las unidades, el rango, la proyección, las dimensiones, el rango valido, entre otros.

```
>>> nc.variables['Rad']
<type 'netCDF4._netCDF4.Variable'>
int16 Rad(y, x)
  _FillValue: 1023
  long_name: ABI L1b Radiances
  standard_name: toa_outgoing_radiance_per_unit_wavelength
  _Unsigned: true
  sensor_band_bit_depth: 10
  valid_range: [ 0 1022]
  scale_factor: 0.8121064
  add_offset: -25.936647
  units: W m-2 sr-1 um-1
  resolution: y: 0.000028 rad x: 0.000028 rad
  coordinates: band_id band_wavelength t y x
  grid_mapping: goes_imager_projection
  cell_methods: t: point area: point
  ancillary_variables: DQF
unlimited dimensions:
current shape = (3000, 5000)
filling on
```

La información del metadato, forma parte de las propiedades del objeto, por lo que los valores son extraídos mediante la colocación de un punto.

```
>>> nc.variables['Rad'].long_name
u'ABIL1bRadiances'
```

Para extraer los **datos de la variable**, se usa la llave dentro del primer corchete del diccionario y se coloca un segundo corchete el cual hace referencia a los datos contenidos.

```
>>> radiancia = nc.variables['Rad'][:]
```

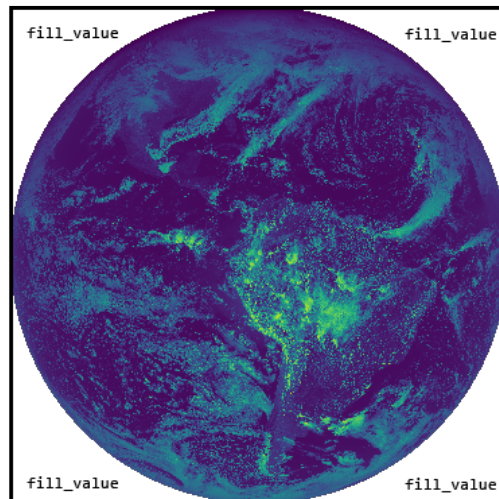
En este caso los valores son asignados a una variable, la cual será un arreglo *Numpy* de tipo enmascarado.

```
>>> print (radiancia)
masked_array(
  data=[[--, --, --, ..., 5.0756454e-02, 5.0756454e-02, 5.0756454e-02, ...
  mask=[[ True,  True,  True, ..., False, False, False, ...
  fill_value=1023,
  dtype=float32)
```

En los arreglos enmascarados los valores numéricos están referenciados a otro valor, generalmente de tipo Boolean<sup>5</sup>.

<sup>5</sup>Puede representar valores de lógica binaria, esto es 2 valores, que normalmente representan falso o verdadero

Sirve para identificar valores no válidos en el arreglo, como el área de las esquinas o valores enmascarados de un producto, como áreas de nubosidad. El parámetro *fill\_value* describe el valor numérico en la matriz de valores numéricos, ver Figura 4.3.



**Figura 4.3:** Valores enmascarados de las esquinas del Full Disk, asociados a valores nulos en el arreglo de datos, elaboración propia con datos de GOES-16.

Para acceder a los valores numéricos de un arreglo enmascarado, se subtrae el parámetro *data*.

```
>>> radiancia.data
array([[1023, 1023, 1023, ..., 5.0756454e-02, 5.0756454e-02, 5.0756454e-02])
```

Este método devuelve los valores en un arreglo *Numpy*, solo con los valores numéricos.

Algunas de las variables de los NetCDF4 de GOES-16 necesitan ser multiplicadas por el **factor de escala** y añadirles un **offset**. Esto se realiza por que los valores contenidos no son los verdaderos, sino un tipo de dato que ocupa menos memoria. Al ser modificadas con estos parámetros se obtiene el valor real. Los parámetros son *valid.range* y *scale.factor* obtenidos como parámetros del objeto.

```
>>> scale = nc.variables['Rad'].scale_factor
>>> offset = nc.variables['Rad'].add_offset
>>> radiancia = radiancia * scale + offset
```

El proceso de extracción de variables también se ocupa para los datos *DQF*, los cuales son arreglos de las mismas dimensiones que la variable principal del producto, ya que contienen el valor de calidad del producto por cada valor.

```
>>> dqf = nc.variables['DQF'][:]
```

Los NetCDF4 contienen además variables anexas con arreglos de diferentes dimensiones, como la variable  $x$  y  $y$ , las cuales son de una sola dimensión. El valor de  $y$  es igual al número de filas y el de  $x$  al número de columnas del arreglo de la variable principal.

```
>>> y = nc.variables['y'][:]  
>>> x = nc.variables['x'][:]
```

Variables anexas con arreglos de un solo valor, contienen datos de constantes físicas, los tiempos de escaneo, las extensiones de la imagen, valores estadísticos, entre otros. Ver Tabla 4.3.

**Tabla 4.3:** Grupos y dimensiones de las variables contenidas en los archivos NetCDF4 de GOES-16.

Grupo	Dimensiones
Datos de la variable principal y de calidad.	Bidimensional
Datos de tiempo.	Unidimensional
Datos de Georreferencia.	Unidimensional y bidimensional
Datos de la banda.	Unidimensional
Datos estadísticos.	Unidimensional
Datos del procesamiento.	Unidimensional

### Datos Suomi-NPP.

Los datos utilizados de nivel L2 del satélite Suomi-NPP fueron adquiridos en formato GeoTIFF a través del servidor de Satélites Polares del LANOT, el cual procesa la señal del satélite por medio de una antena mecánica, la cual se posiciona en la órbita del paso del satélite. Este genera archivos HDF5 a través del software CSPP-Polar. Estos archivos en formato HDF5 son transformados a GeoTIFF por medio del programa Polar2grid Reprojection Software<sup>6</sup>, el cual realiza las correcciones geométricas a los datos así como la reproyección y cambio de formatos.

Los datos utilizados de nivel L2 del satélite Suomi-NPP fueron adquiridos en formato GeoTIFF.

Las características del formato GeoTIFF están descritas en el **Anexo de Formatos**.

Se ocupó el submodulo *GDAL* del módulo *OSGeo* para la extracción de datos. Este módulo permite la lectura de casi todos los formatos raster existentes mediante la función *Open*.

```
>>> from osgeo import gdal  
>>> dataset = gdal.Open('archivo_suomiNPP.tif')
```

<sup>6</sup>Programa para crear fácilmente imágenes reproyectadas de alta calidad, con datos de satélites polares. [https://cimss.ssec.wisc.edu/cspp/npp\\_polar2grid\\_v2.2.shtml](https://cimss.ssec.wisc.edu/cspp/npp_polar2grid_v2.2.shtml)

La importación del módulo contiene todos los controladores de todos los formatos compatibles. La función *Open* reconoce el controlador mediante la extensión del archivo, creando un objeto *gdal dataset*.

Para recuperar los datos de una banda de un GeoTIFF, se requiere crear un objeto de la clase *Band* mediante la función *GetRasterBand*.

```
>>> band = ds.GetRasterBand(1)
```

La variable *band* ahora contiene toda la información de la banda, tanto el conjunto de datos como el metadato. Para la extracción del conjunto de datos se ocupa la función *ReadAsArray*.

```
>>> data = band.ReadAsArray()
```

La variable *data* es un arreglo *Numpy* con los valores del producto.

A diferencia de los archivos NetCDF4, los archivos GeoTIFF con los datos de nivel L2 contiene un metadato menor. Para extraer valores del metadato se aplican las siguientes funciones al objeto *Dataset*.

La función *GetDriver* devolverá el formato del archivo en nombre corto o largo.

```
>>> dataset.GetDriver().ShortName
>>> dataset.GetDriver().LongName
```

Los siguientes parámetros devuelven la resolución y la cantidad de bandas.

```
>>> dataset.RasterXSize
>>> dataset.RasterYSize
>>> dataset.RasterCount
```

Los datos del sistema de proyección y referencia se obtienen con la función *GetProjection*.

```
>>> dataset.GetProjection()
```

Los GeoTIFF contienen una serie de datos contenidos en una tupla<sup>7</sup>, estos son nombrados valores de geo transformación, los cuales definen aspectos del sistema de coordenadas. En una imagen referenciada al norte y sin ninguna rotación o corte, los valores que debe tenerla la tupla de geo transformación son los siguientes.

(xmin, xres, 0, ymax, 0, -yres)

---

<sup>7</sup>Es un conjunto ordenado e inmutable de elementos del mismo o diferente tipo.



Para la obtención de los valores de geo transformación se ocupa la función *GetGeoTransform*.

```
>>> dataset.GetGeoTransform()
```

En la Tabla 4.4 se muestran el ejemplo algunos de los atributos del metadato de un GeoTIFF del producto SST de Suomi-NPP.

**Tabla 4.4:** Atributos del metadato de un GeoTIFF del producto SST de VIIRS/Suomi-NPP.

Atributo del metadato	Valor
dataset.GetDriver().ShortName	'GTiff'
dataset.GetDriver().LongName	"GeoTIFF"
dataset.RasterXSize,	4800
dataset.RasterYSize,	3200
dataset.RasterCount	1
dataset.GetProjection()	GEOGCS["WGS 84", DATUM["WGS 1984", SPHEROID["WGS 84", 6378137, 298.257223563, AUTHORITY["EPSG","7030"]], AUTHORITY["EPSG","6326"]], PRIMEM["Greenwich",0, AUTHORITY["EPSG","8901"]], UNIT["degree",0.01745329251994328, AUTHORITY["EPSG","9122"]], AUTHORITY["EPSG","4326"]]
dataset.GetGeoTransform()	(-115.8075, 0.0078, 0.0, 33.9984, 0.0, 0.0071)

## Datos Sentinel-2.

Los datos utilizados del satélite Sentinel-2 fueron adquiridos por medio de una petición WMS del servidor Sentinel-HUB, el cual proporciona imágenes Sentinel-2 en un registro temporal con fecha en el inicio operacional en 2015. También permite realizar diferente compuesto RGB, búsqueda por porcentaje de nubosidad y aplicar diversas correcciones.

Las peticiones WMS fueron almacenadas en formato PNG. Las características del formato WMS están descritas en el **Anexo de Formatos**.

Para realizar la petición WMS se ocupó el módulo *Requests*. Este módulo esta lanzado bajo la licencia de Apache2<sup>8</sup> y permite hacer solicitudes HTTP<sup>9</sup> de manera más sencilla.

El servicio WMS de Sentinel-HUB fue obtenido de la siguiente dirección *url*.

<https://services.sentinel-hub.com/ogc/wms/>

Los siguientes datos del *url* corresponden a los parámetros del WMS. Estos parámetros están descritos en la Tabla 4.5 con valores de ejemplo.

<sup>8</sup>Contenedor web.

<sup>9</sup>Protocolo de comunicación que permite las transferencias de información por web.

Tabla 4.5: Parámetros modificables de la petición WMS del servidor de Sentinel-HUB.

Parámetro	Descripción
SERVICE=WMS	La petición del servicio.
REQUEST=GetMap	Recupera una imagen de mapa para un área y contenido específicos.
MAXCC=20	Máximo porcentaje de cobertura nubosa.
LAYERS=1-NATURAL-COLOR	Capas para mostrar en el mapa.
EVALSOURCE=S2	Fuente de evaluación.
WIDTH=1286	Ancho del mapa.
HEIGHT=557	Altura del mapa.
FORMAT=image/jpeg	Formato de salida de la solicitud.
NICENAME=	Nombre del archivo.
Sentinel-2+image+on+2019-01-12.jpg	Fecha del inicio del intervalo de búsqueda/Fecha de la imagen.
TIME=2018-07-01/2019-01-12	Fecha del inicio del intervalo de búsqueda/Fecha de la imagen.
BBOX=-399303,4929003,-387207,4934325	Cuadro delimitador para la extensión del mapa. El valor es minx, miny, maxx, maxy en unidades del SRS.

La petición se realiza con la función *get* la cual crea un objeto *Requests*.

```
>>> import requests
>>> url = 'https://services.sentinel-hub.com/ogc/wms/...'
>>> r = requests.get(url)
```

La variable contiene información de la petición. Es común comprobar el estado de la petición con la función *status\_code*, si la clave corresponde al número 200, la solicitud fue aprobada.

```
>>> r.status_code
200
```

Para generar el archivo PNG de la petición WMS, se abre un archivo en modo de escritura binario en el sistema y se escribe en el contenido del WMS.

```
>>> code = open("sentinel2.png", "wb")
>>> code.write(r.content)
```

## 4.2. Tratamiento

En el tratamiento de los datos se usa principalmente el módulo *Numpy* ya que los datos extraídos del NetCDF4 de GOES-16 y del GeoTIFF de Suomi-NPP son ahora arreglos matriciales.

El tratamiento con los datos de Sentinel-2 es nulo, ya que las peticiones WMS ya tienen un tratamiento realizado desde el servidor de adquisición.

### Datos Goes-16 y Suomi-NPP.

El recorte de los arreglos *Numpy* se realiza bajo un esquema matricial en donde las comas dentro del corchete dividen las dimensiones y los dos puntos el intervalo, siendo la primera el intervalo de filas y el segundo el intervalo de columnas para los datos del arreglo.

```
>>> dataRec = data[500:1000,1000:1500]
```

Es de utilidad cuando no se requiere usar todos los datos de la variable, ver Figura 4.4.

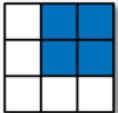

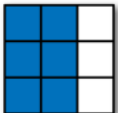

	Expresión	Forma
	<code>arr[:2, 1:]</code>	(2, 2)
	<code>arr[2]</code> <code>arr[2, :]</code> <code>arr[2:, :]</code>	(3,) (3,) (1, 3)
	<code>arr[:, :2]</code>	(3, 2)
	<code>arr[1, :2]</code> <code>arr[1:2, :2]</code>	(2,) (1, 2)

Figura 4.4: Expresiones y forma de selección en arreglos *Numpy*, tomada de <https://forums.fast.ai/t/unofficial-lesson-8-classnotes/8184>.

Los arreglos *Numpy* pueden ser rescalados mediante la función *reshape*, el cual cambia las dimensiones del arreglo. Esto es necesario cuando se van a realizar operaciones entre bandas de diferente resolución espacial.

En la Tabla 4.6 se muestra las dimensiones de los arreglos según la resolución espacial, para las regiones de obtención de GOES-16 y Suomi-NPP.

Tabla 4.6: Dimensiones de los arreglos de datos de GOES-16 y Suomi-NPP (Región utilizada en color gris).

Región.	Resolución Espacial.	Dimensión.
GOES-16		
Fulldisk.	0.5 km, 1 km, 2 km	(21696, 21696), (10848, 10848), (5424, 5424)
CONUS.	0.5 km, 1 km, 2 km	(6000, 10000), (3000, 5000), (1500, 2500)
Mesoescala.	0.5 km, 1 km, 2 km	(2000, 2000), (1000, 1000), (500, 500)
Suomi-NPP		
Región de paso para México.	0.75 km	(3200, 4800)

La función *reshape* condiciona a que las nuevas dimensiones deben de poder almacenar el mismo número de elementos. Por ejemplo, el arreglo de una escena en la región CONUS a 1 km tiene 3000 filas y 5000 columnas, por lo tanto tiene 15000000 de valores. Una redimensión válida para este arreglo sería (1000,15000).

```
>>> np.reshape(data,(1000,15000))
```

Para eliminar valores y redimensionar a cualquier valor, es necesaria una función más elaborada basada en la función *reshape*, la cual está descrita en el **Anexo de Scripts** con el nombre de *rebin*. La redimensión solo puede hacerse de valores mayores a menores.

Los arreglos *Numpy* pueden ser operados matemáticamente entre ellos de manera similar a como se hace con números enteros y flotantes. Estos deben ser compatibles en el tamaño, ya que las operaciones a cada valor del arreglo se realizan con el valor del otro arreglo en la misma ubicación, tomando en cuenta el número de fila y columna, ver Figura 4.5.

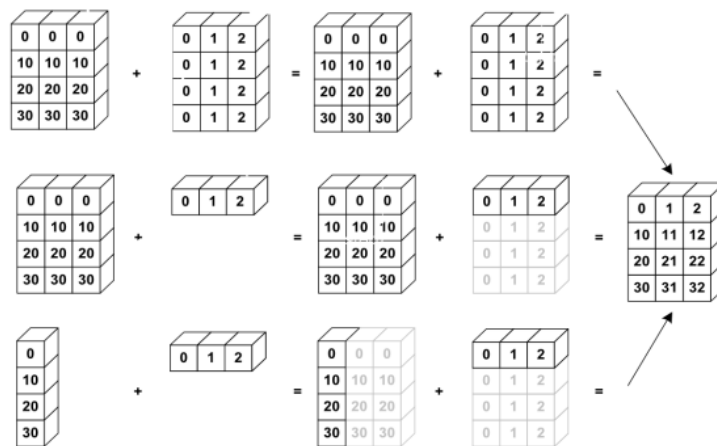


Figura 4.5: Operaciones compatibles en arreglos *Numpy*, tomada de <https://www.iodocs.com/difference-reshape-resize/>.

Por ejemplo si se quiere obtener el NDVI<sup>10</sup> con la banda 2 que está en el rojo y la banda 3 que está en el NIR, de las bandas del sensor ABI de GOES-16 en la región de FullDisk, es necesario realizar una redimensión de los valores de la banda 2 que está a 0.5 km, a las dimensiones de la banda 3 que está a 1 km.

```
>>> from netCDF4 import Dataset
>>> import rebin
>>> nc2 = Dataset('OR_ABI-L1b-RadF-M3C02-G16-s20183640002188_e20183640004561
_c20183640005004.nc', 'r')
>>> nc3 = Dataset('OR_ABI-L1b-RadF-M3C03-G16-s20183640002188_e20183640004561
```

<sup>10</sup>Índice que se utiliza para estimar la cantidad, calidad y desarrollo de la vegetación con base a la medición de la intensidad de la radiación de ciertas bandas del espectro electromagnético.

```

_c20183640005004.nc', 'r')
>>> radiancia2 = nc2.variables['Rad'][:]
>>> radiancia3 = nc3.variables['Rad'][:]
>>> rebin(radiancia2, (10848, 10848))
>>> NDVI = radiancia3 - radiancia2 / radiancia3 + radiancia2

```

El uso de la función importada *rebin*, es fundamental para realizar operaciones matemáticas entre bandas con diferente resolución de una misma región.

Los valores de los arreglos pueden ser reasignados mediante el uso del valor de sus índices, ya que hacen referencia a la ubicación en la que se encuentran, ver Figura 4.6.

		Eje 1		
		0	1	2
Eje 0	0	0,0	0,1	0,2
	1	1,0	1,1	1,2
	2	2,0	2,1	2,2

**Figura 4.6:** Ubicación de valores en arreglos *Numpy*, tomada de <https://docplayer.net/8946162-Python-for-data-analysis-wes-mckinney.html>.

La reasignación en listas se suele realizar con el uso de sentencias de control como los ciclos *for*, en los cuales se recorre el arreglo elemento a elemento y se reasigna el valor con un condicional.

Por ejemplo, en el arreglo del NDVI calculado anteriormente, para reasignar a 0 los elementos cuyo valor es menor a 0 sin hacer uso de *Numpy*, el proceso tendría que recorrer el arreglo con un doble ciclo *for*, colocando la condición y reasignando los valores.

```

>>> for i in NDVI:
>>>     for j in NDVI:
>>>         if NDVI[i][j] < 0:
>>>             NDVI[i][j] = 0

```

Sin embargo el proceso es más tardado pues recorre la matriz elemento a elemento. En arreglos *Numpy* se puede poner la condicional dentro del corchete y definir el valor de reasignación.

```

>>> NDVI [NDVI < 0] = 0

```

Una de las asignaciones regulares es la de los valores *no data*, ya que en determinados procesos fue necesario realizar esta asignación de forma manual. El valor *no data* es un tipo de valor único al que no corresponde ningún valor. Por ejemplo, si se tuviera que colocar un valor *no data* en vez de 0 en la reasignación de los valores del NDVI menores a 0, se tendría que realizar de esta forma.

```
>>> import numpy as np
>>> NDVI [NDVI < 0] = np.nan
```

Una función similar a la reasignación en *Numpy*, es la función *where*, este devuelve una tupla con dos listas de los índices de los valores en el arreglo que cumple una condición. Las tuplas son parecidas a las listas, solo que estas son inmutables.

Se podría decir que la función *where* es un buscador de posiciones en el arreglo, el cual devuelve pares ordenados de posiciones en el arreglo. Por ejemplo si se quisieran encontrar la posición de los valores del NDVI que están entre 0.5 y 0.6, se tendría que realizar de la siguiente forma.

```
>>> np.where(NDVI > 0.5 and NDVI < 0.6)
[[67,435,2330,5764],[345,1234,670,654,33]]
```

De los arreglos *Numpy* se pueden obtener datos estadísticos de una manera inmediata, como el valor máximo, el mínimo, la media, la sumatoria, entre otros.

```
>>> data.max()
0.73
>>> data.min()
0.0
>>> data.mean()
0.35
>>> data.sum()
12.34
```

Para la creación de composiciones RGB con varios arreglos *Numpy*, se ocupó la función *dstack*, la cual crea un nuevo arreglo *Numpy* que contiene los arreglos agregados a la función. Es importante convertir los valores de los arreglos a enteros, ya que estos serán asignados a colores.

```
>>> rgb = np.dstack((r,g,b)).astype('uint8')
```

Estas funciones y métodos del módulo *Numpy* fueron los más usados en los procesos a los arreglos de datos de GOES-16 y Suomi-NPP.

### 4.3. Georreferencia

Los métodos de georreferencia fueron realizados de acuerdo a la información del sistema de referencia y proyección con la que se contaba en cada formato.

Se describen cuatro métodos diferentes ocupados en los procesos.

- **Método 1.** Se aplicó solo a los NetCDF4 de GOES-16 ya que ocupa las coordenadas contenidas como variables. Utiliza el módulo *Numpy* y el de *NetCDF4*.
- **Método 2.** Puede ser aplicado tanto a los datos de NetCDF4 como GeoTIFF, ya que se basa en la creación de un formato raster georreferenciado a partir de los arreglos *Numpy*. Ocupa el módulo de *Numpy* y el de *GDAL*.
- **Método 3.** Es exclusivo para las peticiones WMS de Sentinel-2, es el más corto ya que solo necesita de cambiar el parámetro del cuadrante de la petición.

Todos los métodos fueron utilizados en los procesos.

#### Método 1. Datos GOES-16.

Este método se basa en la utilización de la información de georreferencia contenida en el NetCDF4 para asociar a las variables principales con su sistema de referencia original.

Los datos de GOES-16 son obtenidos en un sistema de proyección geoestacionaria, la cual necesita de ciertos parámetros para su definición en las distintas bibliotecas de sistemas de referencia. Los parámetros están descritos en Tabla 4.7 con la clave usada en el formato PROJ4<sup>11</sup>.

La altura satelital es un parámetro obligatorio en la proyección geoestacionaria, ya que las coordenadas se relacionan con el ángulo de exploración mediante la siguiente expresión:

$$A = \frac{P}{h} \quad (4.1)$$

Dónde:

$A$  = Ángulo de escaneo (radianes).

$P$  = Coordenada proyectada.

<sup>11</sup>PROJ4 es una biblioteca para realizar conversiones entre proyecciones cartográficas, el formato PROJ4 hace referencia a un proj-string, en cual describe cualquier transformación sin importar cuán simple o complicada pueda ser

Tabla 4.7: Parametros de georreferencia de los datos de GOES-16 en el formato PROJ4.

Parámetro	Clave	Valor
<b>Obligatorios</b>		
Nombre de la proyección.	+proj	geos
Altura satelital o del punto de perspectiva.	+h	35786023.0
<b>Opcionales</b>		
Eje del ángulo de barrido del instrumento.	+sweep	x
Longitud del centro de proyección.	+lon_0	-75.2
Elipsoide de referencia.	+ellps	GRS80
Falso este.	+x_0	0
Falso norte.	+y_0	0

$h$  = Altura satelital del punto de perspectiva.

Las coordenadas proyectadas se obtienen al despejar  $P$  de la ecuación 4.1.

$$P = Ah \quad (4.2)$$

Para conocer el valor de  $h$  directamente de la variable  $nc$ , se accede a la variable `goes_imager_projection`, en la cual está contenida toda la información de la proyección de los datos de GOES-16.

```
>>> nc.variables['goes_imager_projection']
<type 'netCDF4._netCDF4.Variable'>
int32 goes_imager_projection()
  long_name: GOES-R ABI fixed grid projection
  grid_mapping_name: geostationary
  perspective_point_height: 35786023.0
  semi_major_axis: 6378137.0
  semi_minor_axis: 6356752.31414
  inverse_flattening: 298.2572221
  latitude_of_projection_origin: 0.0
  longitude_of_projection_origin: -75.0
  sweep_angle_axis: x
unlimited dimensions:
current shape = ()
filling on, default _FillValue of -2147483647 used
```

Los datos adicionales que contiene la variable de proyección, son necesarios para poder definirla en el proceso de mapeo.

Adicionalmente, las variables `nominal_satellite_subpoint_lat`, `'nominal_satellite_subpoint_lon'` y `'nominal_satellite_height'`, contienen los mismos valores que las propiedades `latitude_of_projection_origin`, `longitude_of_projection_origin` y



perspective\_point\_height de la variable 'goes\_imager\_projection', salvo que el valor de longitud de origen es -75.2 y la altura satelital está en km en las variables.

Es más recomendable usar los arreglos unidimensionales de las variables, que las propiedades de proyección, ya que los arreglos son variables del NetCDF4 y pueden cambiar su valor.

Los valores de A de la ecuación están contenidos en las variables X y Y. Estas variables contienen los principales datos de georreferencia, ya que son arreglos que contienen los valores de los ángulos de escaneo en el sistema de proyección de GOES-16 para cada elemento de los datos principales. Si se accede a la variable, el parámetro de unidades aparece en radianes.

```
>>> nc.variables['x']
<type 'netCDF4._netCDF4.Variable'>
int16 x(x)
  scale_factor: 2.8e-05
  add_offset: -0.101346
  units: rad
  axis: X
  long_name: GOES fixed grid projection x-coordinate
  standard_name: projection_x_coordinate
unlimited dimensions:
current shape = (5000,)
filling on, default _FillValue of -32767 used
```

Su extracción se realiza de la misma forma que las variables anteriores. Se multiplica por el factor de escala y se le añade el offset.

```
>>> scale = nc.variables['x'].scale_factor
>>> offset = nc.variables['x'].add_offset
>>> X = nc.variables['x'][:] * scale + offset
>>> scale = nc.variables['y'].scale_factor
>>> offset = nc.variables['y'].add_offset
>>> Y = nc.variables['y'][:] * scale + offset
```

Para convertirlas en coordenadas geostacionarias, se multiplican por la altura del punto de perspectiva<sup>12</sup> en metros.

```
>>> h = nc.variables['nominal_satellite_height'][:]*1000
>>> lons = X*h
>>> lats = Y*h
```

Las variables *lons* y *lats* ahora contienen las coordenadas proyectadas de cada elemento. Es necesario crear una matriz de coordenadas si se quieren utilizar para operar los datos de las variables, ya que son arreglos unidimensionales. Esto se realiza por medio de la función *meshgrid* de *Numpy*, la cual devuelve una matriz de coordenadas de

<sup>12</sup>Altura satelital.

los vectores de  $X$  y  $Y$ .

```
>>> XX, YY = np.meshgrid(X, Y)
```

Las variables  $XX$  y  $YY$  pueden ser utilizadas en asociación con los datos de algunas variables para el mapeo de la información.

### **Método 2. Datos GOES-16 y Suomi-NPP.**

Este método consiste en georreferenciar los datos a través de las coordenadas extremas de la región de escaneo y la resolución espacial.

En los GeoTIFF de Suomi-NPP estos valores ya están contenidos en la tupla de geotransformación obtenida con la función *GetGeoTransform* de **GDAL**.

En el caso de los archivos NetCDF4 de GOES-16 las coordenadas extremas se encuentran en las variables 'y\_image\_bounds' y 'x\_image\_bounds'.

Las coordenadas extremas de GOES-16 corresponden a valores de escaneo, por lo cual se encuentran en radianes y necesitan ser multiplicadas por la altura del punto de perspectiva.

```
>>> x1 = nc.variables['x_image_bounds'][0] * h
>>> x2 = nc.variables['x_image_bounds'][1] * h
>>> y1 = nc.variables['y_image_bounds'][1] * h
>>> y2 = nc.variables['y_image_bounds'][0] * h
```

La resolución espacial de los datos se puede obtener con la siguiente fórmula.

$$r = \frac{(x_{max} - x_{min})}{nx} \quad (4.3)$$

Dónde:

$r$  = resolución.

$x_{max}$  = coordenada máxima.

$x_{min}$  = coordenada mínima.

$nx$  = dimensión de las coordenadas

El valor de  $nx$  en la Ecuación 4.3 se puede calcular con el parametro *shape* del arreglo **Numpy**, ya que se refiere al número de elementos dentro del arreglo. Este devuelve una lista de dos elementos en el caso de las matrices de dos dimensiones, siendo el primer valor el número de filas y el segundo el de columnas.

```
>>> nx = data.shape[0]
>>> ny = data.shape[1]
```

La resolución se calcula para  $x$  y  $y$ .

```
>>> xres = (xmax - xmin) / ny
>>> yres = (ymax - ymin) / nx
```

Estos datos permiten la georreferencia de un arreglo *Numpy* mediante la creación de un raster. El raster es creado a través de los módulos *GDAL* y *OSR* del módulo *OSGeo*.

Para realizar lo anterior se debe de crear una variable que contenga el controlador del formato raster que se quiere crear. Un método sencillo es realizarlo por el nombre corto del formato con la función *GetDriverByName* y definir los parámetros en la función *Create*. Los parámetros para un GeoTIFF son nombre del archivo, la resolución del pixel, número de bandas y el tipo de dato.

```
>>> dst_ds = gdal.GetDriverByName('GTiff').Create('tmp.tif', ny, nx, 1, gdal.GDT_Float32)
```

La variable *dst\_ds* ahora tiene el controlador necesario para la creación de un GeoTIFF. Para colocar los datos de la geotransformación, se utiliza la función *SetGeoTransform*.

```
>>> dst_ds.SetGeoTransform(geotransform)
```

El raster se ha creado con el sistema de coordenadas y el número de pixeles, pero no tiene sistema de referencia espacial ni valores en sus pixeles. Para definir el sistema de referencia se usa el módulo *OSR* del módulo *OSGeo*, el cual está diseñado para manejo de proyecciones y sistemas de referencia.

Se debe de crear una variable de referencia espacial de la clase *spatial reference* e importar el sistema de referencia en base a algún formato conocido.

En la Tabla 4.8 se muestra el sistema de referencia para GOES-16, en tres de los formatos de referencia compatibles con *OSR*.

El formato de sistema de referencia usado fue el PROJ4 por su simplicidad.

Tabla 4.8: Sistema de referencia de los datos de GOES-16 en formatos compatibles con *OSR*.

Formato	Formato para GOES-16
Human-Readable OGC WKT o prj	PROJCS['unnamed', GEOGCS['unknown', DATUM['unknown', SPHEROID['GRS80',6378137,298.2572221]], PRIMEM['Greenwich',0], UNIT['degree',0.0174532925199433]], PROJECTION['Geostationary_Satellite'], PARAMETER['central_meridian',-75.2], PARAMETER['satellite_height',35786023.0], PARAMETER['false_easting',0], PARAMETER['false_northing',0]
PROJ4	+proj=geos +lon_0=-75.2 +h=35786023 +x_0=0 +y_0=0 +ellps=GRS80 +units=m +no_defs +sweep=x

```
>>> from osgeo import osr
>>> srs = osr.SpatialReference()
>>> srs.ImportFromProj4("+proj=geos +h=35786023.0 +ellps=GRS80 +lat_0=0.0 +lon_0=-75.0  
+sweep=x +no_defs")
```

Para colocar el sistema de referencia en el GeoTIFF se usa la función *setProjection*, la cual requiere como parámetro un objeto de la clase *spatial reference*, el objeto está contenido en la variable *srs* con la referencia espacial de GOES-16.

El formato de referencia del GeoTIFF se mantendrá en los distintos visualizadores de metadatos, por lo que es conveniente colocar un formato más legible del sistema de referencia, como el WKT. Esto se realiza con la función *ExportToWKT* de la clase *spatial reference*.

```
>>> dst_ds.SetProjection(srs.ExportToWKT())
```

Ahora el GeoTIFF contiene el sistema de referencia y de proyección, así como las dimensiones y la resolución de los pixeles, pero los pixeles aún se encuentran sin valor.

La asignación de valores se hace en referencia al número de banda. Los GeoTiff pueden almacenar distintas bandas en un mismo archivo. Con la función *GetRasterBand* se coloca el número de la banda a procesar de la variable *dst\_ds*, en este caso solo existe una. Con la función *WriteArray* se escriben los valores contenidos en el arreglo *Numpy*.

```
>>> dst_ds.GetRasterBand(1).WriteArray(data)
```

Finalmente se ocupa la función *FlushCache* que se encarga de vaciar todos los datos en caché de escritura en el disco, ya que cualquier dato raster escrito a través de *GDAL*, pero almacenado temporalmente en el búffer, se escribirá en el disco. También es conveniente declarar la variable vacía.

```
>>> dst_ds.FlushCache()
>>> dst_ds = None
```

El GeoTIFF es creado en el mismo directorio que el script de Python, si no es que no se coloca ninguna ruta de directorios anterior al nombre. Este GeoTIFF contiene los valores del arreglo *Numpy* y está georreferenciado de manera correcta.

### Método 3 Sentinel-2.

El método de georreferencia para las peticiones WMS de Sentinel-2 consistió en la modificación de los valores del parámetro BBOX el cual define las extensiones de la imagen.

La obtención de los valores mínimos y máximos se realizó con la creación de la función *CoordenadasVentana* el cual se encuentra en el **anexo de scripts**. Esta función tiene como parámetro un punto central, al cual se le suma la extensión en las unidades de la proyección.

El punto central es colocado como una coordenada geográfica (latitud, longitud), con código de sistema de referencia EPSG:4326<sup>13</sup> para facilitar su funcionamiento en los procesos. Los valores extremos obtenidos de la función *CoordenadasVentana*, fueron proyectados con el módulo *PyProj*, el cual realiza transformaciones cartográficas entre coordenadas geográficas (lat/lon), coordenadas proyectadas (x/y) y también entre diferentes sistemas de proyección.

En la Tabla 4.9 se muestra el sistema de referencia para el WMS de Sentinel-2 en tres de los formatos de referencia compatibles con *OSR*.

**Tabla 4.9:** Sistema de referencia para las peticiones WMS de Sentinel-2 en formato EPSG, PROJ4 y PRJ.

Formato	Formato para GOES-16
EPSG	3857
PROJ4	+proj=merc +a=6378137 +b=6378137 +lat_ts=0.0 +lon_0=0.0 +x_0=0.0 +y_0=0 +k=1.0 +units=m +nadgrids=@null +wktext +no_defs
.prj	PROJCS["WGS84/Pseudo-Mercator", GEOGCS["GCS.WGS.1984", DATUM["D.WGS.1984", SPHEROID["WGS.1984", 6378137,298.257223563]], PRIMEM["Greenwich",0], UNIT["Degree", 0.017453292519943295]], PROJECTION["Mercator"], PARAMETER[central_meridian",0], PARAMETER["scale_factor",1], PARAMETER["false_easting",0], PARAMETER["false_northing",0], UNIT["Meter",1]]

Se ocuparon los códigos EPSG para definir los parámetros del sistema de proyección de la coordenada de entrada y la del sistema de salida mediante la función *Proj*.

La transformación de las coordenadas se realiza con la función *transform*.

<sup>13</sup>EPSG es un repositorio de parámetros geodésicos que contiene información sobre sistemas de referencia antiguos y modernos, proyecciones cartográficas y elipsoides de todo el mundo.

```
>>> from pyproj import Proj, transform
>>> lllon, lllat, urlon, urlat = coordenadasVentana(x,y,offset)
>>> lllonMer, lllatMer = transform(Proj(init='epsg:4326'), Proj(init='epsg:3857'), lllon, lllat)
>>> urlonMer, urlatMer = transform(Proj(init='epsg:4326'), Proj(init='epsg:3857'), urlon, urlat)
```

Las variables obtenidas son asignadas al parámetro BBOX, obteniendo una imagen georreferencia al punto central colocado.

## 4.4. Reproyección

Cuando a los datos se les ha definido su sistema de proyección y referencia, estos pueden ser reproyectados a cualquier otro sistema.

Se describen dos métodos utilizados en los procesos.

- **Método 1.** Se aplicó solo a los NetCDF4 de GOES-16, ya que ocupa los arreglos de las variables X y Y. Utiliza el módulo *PyProj* y *Numpy*.
- **Método 2.** Puede ser aplicado a los datos de GOES-16, Suomi-NPP y Sentinel-2, ya que utiliza la función de reproyección de raster del modulo *GDAL*.

### Método 1 GOES-16.

En este método se transformaron las coordenadas en sistema de proyección geostacionaria a coordenadas en sistema geográfico.

Se debe de crear una variable con un objeto de la clase *Proj*, con los parámetros de la proyección de GOES-16.

```
>>> from pyproj import Proj
>>> p = Proj(proj='geos', h=35786023.0, lat_0=0, lon_0=-75.0, sweep=x, ellps='GRS80')
```

La variable *p* contiene la definición del sistema de proyección, si se le agregan las variables *XX* y *YY* calculadas anteriormente, las cuales contienen los valores de las coordenadas geostacionarias de forma matricial, *PyProj* transformara sus valores a coordenadas geográficas si se le añade el parámetro *inverse*.

```
>>> lons, lats = p(XX, YY, inverse=True)
```

## Método 2 GOES-16, Suomi-NPP y Sentinel-2.

Este método de reproyección consiste en usar la función *gdalwarp* del módulo **GDAL**. Este comando permite reproyectar rasters a cualquier proyección compatible.

Para reproyectar el GeoTIFF, se anexa el parámetro *dstSRS* a las opciones de la función *Warp*, por medio de la clase *WarpOptions*, este define la proyección de salida mediante códigos de sistemas de referencia, como EPSG, PROJ4 o un archivo PRJ<sup>14</sup>.

```
>>> ds = gdal.Open('tmp.tif')
>>> gdal.Warp('tmp_4326.tif', ds, options=gdal.WarpOptions(dstSRS='EPSG:4326'))
```

En este caso el resultado de salida es un GeoTIFF en un sistema de coordenadas geográfico, según el código EPSG:4326.

## 4.5. Mapeo

El proceso es similar al aplicado a los datos de GOES-16, Suomi-NPP y Sentinel-2, ya que los datos de los tres fueron reproyectados con el mismo sistema de referencia.

Para el mapeo de los datos se necesitaron las extensiones de la imagen.

### GOES-16, Suomi-NPP y Sentinel-2.

Se ocupó el módulo **CartoPy** que forma parte de las extensiones del módulo **Matplotlib**, por lo cual usa funciones de **Matplotlib**, pero además es un módulo para trazar datos y generar mapas en 2D.

Para generar un mapa se crea una variable *axes* de la clase *pyplot* de **Matplotlib**, la cual permite agregar todos los elementos de una figura de *pyplot*. El sistema de proyección es definido con el parámetro del mismo nombre, el cual otorgará las cualidades de la proyección a los elementos mostrados en la imagen.

Las proyecciones se encuentran en la clase *crs* de **CartoPy**. La proyección equivalente al sistema geográfico con código EPSG:4326, es la proyección PlateCarree. Los parámetros de la proyección pueden ser definidos dentro de la función.

```
>>> import cartopy.crs as ccrs
>>> import matplotlib.pyplot as plt
>>> ax = plt.axes(projection=ccrs.PlateCarree())
```

<sup>14</sup>PRJ es el archivo que almacena información del sistema de coordenadas; se utiliza en ArcGIS.

Esto devuelve el espacio de trabajo en el sistema de coordenadas con la proyección asignada. Para mostrar una línea de costa se usa la función *coastlines* del objeto *axes*, la cual descarga un archivo vectorial de un servidor ya definido.

```
>>> ax.coastlines()
```

Para mostrarlo todo en pantalla, se usa la función *show* del submódulo *pyplot* de **Matplotlib**.

```
>>> plt.show()
```

La extensión de la imagen mostrada se ajustará de acuerdo a los elementos adicionales presentes. Para definir una extensión específica con coordenadas del sistema de proyección, se usa la función *set\_extent*.

```
>>> ax.set_extent([-120,-84,34,12])
```

Para la definición del sistema de referencia, se ocupa la clase *Globe*, en la cual se puede definir el elipsoide, semieje mayor, semieje menor, achatamiento y datum. Las Figuras 4.7, 4.8 y 4.9 muestran la creación de un objeto *axes*, con la definición de los tres sistemas de proyección iniciales de los datos ocupados de GOES-16, Suomi-NPP y Sentinel-2 usando **CartoPy**.

```
>>> ax=plt.axes(projection=crrs.Geostationary(central_longitude=-75.0,
satellite_height=35786023.0,globe=crrs.Globe(ellipse='GRS80')))
```



**Figura 4.7:** Mapa global generado con **CartoPy** a partir de los parámetros de proyección de los datos de GOES-16, elaboración propia.

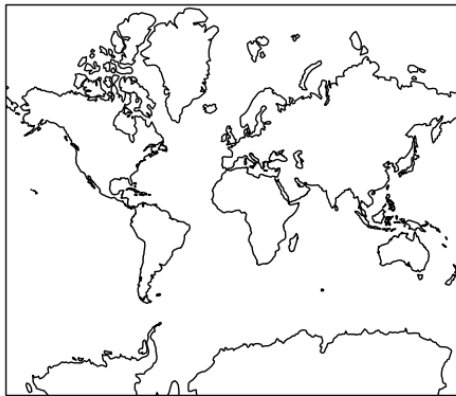
```
>>> ax=plt.axes(projection=crrs.PlateCarree(central_longitude=0.0,
globe=crrs.Globe(ellipse='WGS84',datum='WGS84')))
```





**Figura 4.8:** Mapa global generado con *CartoPy* a partir de los parámetros de proyección de los datos de Suomi-NPP, elaboración propia.

```
>>> ax=plt.axes(projection=ccrs.Mercator(central_longitude=0.0,
globe=ccrs.Globe(ellipse='WGS84',datum='WGS84')))
```



**Figura 4.9:** Mapa global generado con *CartoPy* a partir de los parámetros de proyección de los datos de Sentinel-2, elaboración propia.

En todos los procesos, los mapas finales fueron creados con el código de referencia EPSG:4326.

Para agregar datos al mapa, se ocuparon las funciones del submódulo *pyplot*. En el caso de la colocación de puntos al mapa, se ocupó la función *plot*, la cual tiene como parámetro obligatorio, una coordenada X, una Y y un tipo de marcador definido con claves para el color y forma.

```
>>> import matplotlib.pyplot as plt
>>> plt.plot(x,y,'r+')
```

Para la colocación de los datos en arreglos *Numpy* se ocupa la función *imshow*, la cual permite mostrar imágenes a partir de arreglos.

```
>>> plt.imshow(data, cmap = 'jet', extent=[-120,-84,34,12])
```

El parámetro *cmap* hace referencia a la paleta de colores utilizada.

Para una correcta georreferencia de los datos de la imagen, es necesario colocar las coordenadas de la extensión de la matriz por medio del parámetro *extent*, ya que los arreglos no tienen georreferencia. *CartoPy* permite la importación directa de shapefiles y de complementos como gradículas, etiquetas, texto, imágenes anexas, barras de simbología, enmascaramiento de continentes y océanos, isolíneas, simbología de vectores, mapas base, entre otros.

## 4.6. Procesos

Los procesos utilizaron los productos satelitales de variables meteorológicas y compuestos RGB de cada satélite, agrupando la información de los datos de GOES-16, Suomi-NPP y Sentinel-2 de manera automática.

Los procesos realizados fueron:

- **Proceso 1.** Estaciones meteorológicas EMAS, GOES-16 y Sentinel-2.
- **Proceso 2.** Boyas meteorológicas NOAA, Suomi-NPP y Sentinel-2.
- **Proceso 3.** Puntos de calor, GOES-16, Suomi-NPP y Sentinel-2.

En la Tabla 4.10 se muestra de cuáles satélites provienen los productos satelitales empleados en los procesos.

**Tabla 4.10:** Distribución de productos e intervalo de tiempo de cada satélite en los procesos.

		Productos satelitales.					Intervalo
		Variables meteorológicas.		RGB			
		LST	SST	TC	FT	SWIR	
Proceso 1	GOES-16	*	-	-	-	-	Cada hora
	Suomi-NPP	-	-	-	-	-	
	Sentinel-2	-	-	*	-	-	
Proceso 2	GOES-16	-	-	-	-	-	A las 12:00.
	Suomi-NPP	-	*	-	-	-	
	Sentinel-2	-	-	*	-	-	
Proceso 3	GOES-16	-	-	*	*	-	Cada hora de 10:00 a 18:00
	Suomi-NPP	-	-	*	*	-	
	Sentinel-2	-	-	*	-	*	

En la Tabla 4.11 se muestra el tiempo de adquisición de cada producto de acuerdo al satélite.

Los tres procesos están basados en la localización de una coordenada, en el caso de las estaciones y boyas esta

**Tabla 4.11:** Tiempo de adquisición de cada producto de acuerdo al satélite.

	Productos satelitales.				
	Variables meteorológicas.		RGB		
	LST	SST	TC	FT	SWIR
GOES-16	1 hora	1 hora	5 minutos	5 minutos	-
Suomi-NPP	1 día	1 día	1 día	1 día	-
Sentinel-2	-	-	5 días	-	5 días

coordinada está referida a la ubicación, la cual permanece estática. La coordenada en los recortes de la ubicación de puntos de calor son dinámicas en cada inicio del proceso.

#### 4.6.1. Proceso 1. Estaciones meteorológicas EMAS, GOES-16 y Sentinel-2

El proceso integra los datos del producto LST del sensor ABI/GOES-16, con los datos de estaciones EMAS del SMN, añadiendo el compuesto RGB en TC del último paso de Sentinel-2 en la ubicación de la estación, con los datos sobrepuestos de LST del sensor ABI/GOES-16.

##### Datos de Entrada.

- NetCDF4 del producto de LST del sensor ABI/GOES-16.
- XLS con registros de las últimas 24 hrs de la EMA.
- WMS del compuesto RGB en TC del último paso de Sentinel-2.

##### Datos de salida.

Gráficas con los datos de LST de GOES-16 y temperatura del aire en las últimas 24 horas, en la ubicación de la estación.

Compuesto RGB en TC de Sentinel-2 con los datos sobrepuestos de LST de GOES-16 en un cuadrante aproximado de 5 km.

##### Procesamiento de datos EMAS.

Los datos de las EMAS son obtenidos del servidor de “*Mapa de Estaciones Meteorológicas Automáticas (EMAS)*”, ver Figura 4.10.

El *url* de conexión a los datos de descarga es el siguiente.

<http://smn1.conagua.gob.mx/emas/>

El último parámetro de la dirección define el identificador de la estación, el tiempo de los datos y el formato a partir de la extensión.

[http://smn1.conagua.gob.mx/emas/exc/QR01\\_24H.xls](http://smn1.conagua.gob.mx/emas/exc/QR01_24H.xls)

Se ocuparon los datos de las últimas 24 hrs en formato XLS de 32 estaciones, una por cada estado. Estas estaciones fueron elegidas por no presentar problemas de transmisión de datos y cubrir el territorio mexicano. El identificador de las estaciones fue almacenado en una variable de tipo lista.

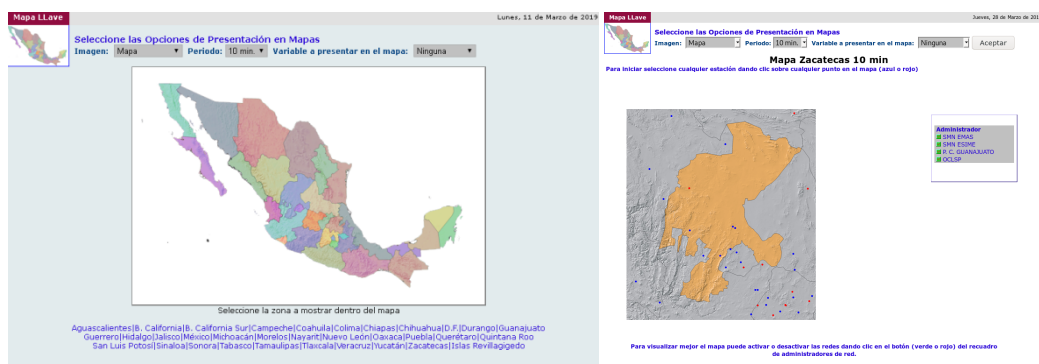


Figura 4.10: Página del SMN con el mapa de Estaciones Meteorológicas Automáticas (EMAS), tomada de <http://smn1.conagua.gob.mx/emas/>.

La descarga se realizó por medio del módulo *urllib3*, a partir de un ciclo *for* que recorre la variable de estaciones. En cada recorrido se descarga el archivo, se extrae los datos, se grafican y se eliminan los archivos.

Para la extracción de los datos en los archivos XLS se ocupó el módulo *Pandas*, para transformar a formato CSV y crear un *Dataset*. Los datos de temperatura y tiempo de registro fueron localizados mediante la identificación del nombre de los atributos.

Las coordenadas de cada estación fueron extraídas del encabezado de cada archivo, ver Figura 4.11.

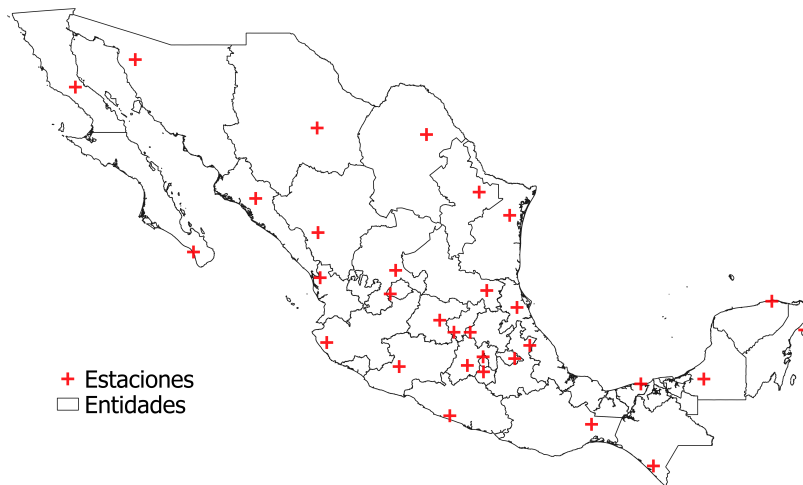


Figura 4.11: Estaciones EMAS seleccionadas, elaboración propia.

### Procesamiento de datos LST de GOES-16.

Para la extracción de los datos del producto LST del sensor ABI de GOES-16 en la coordenadas de las estaciones EMAS, se creo una función denominada *Buscador*, la cual se encuentra en el **anexo de scripts**.

La función *Buscador* trabaja con datos en formato GeoTIFF, con un sistema de referencia EPSG:4326, por lo que se crearon GeoTIFF temporales a partir de los datos del NetCDF4 de LST de GOES-16.

El script retorna el valor del producto en la ubicación de la coordenada mediante la función secundaria *buscador-TIFF*, la cual se encuentra en el **anexo de scripts**. Esta función obtiene la fila y columna del pixel asociado a la coordenada del sistema, mediante la siguiente relación:

$$F = \frac{(X - X_{min})}{r_x} \quad (4.4)$$

$$C = \frac{(Y - Y_{min})}{r_y} \quad (4.5)$$

Donde:

$F$  = Índice de fila buscada.

$C$  = Índice de columna buscada.

$X$  = Coordenada X buscada.

$Y$  = Coordenada Y buscada.

$X_{min}$  = Coordenada X mínima.

$Y_{min}$  = Coordenada Y mínima.

$r_x$  = Resolución del pixel en X.

$r_y$  = Resolución del pixel en Y.

La relación solo es válida para coordenadas geográficas en el cuadrante Norte-Este, donde las longitudes son negativas y las latitudes positivas.

Los valores requeridos fueron obtenidos de la tupla de geotransformación.

```
>>> geotransform = ds.GetGeoTransform()
>>> xRes = transform[1]
>>> yRes = -transform[5]
>>> xMin = transform[0]
>>> yMin = transform[3] - data.shape[1]*yRes
>>> f = int((x - xMin) / xRes)
>>> c = int((y - yMin) / yRes)
```

El valor de fila y columna corresponde a los índices localizados en el arreglo, por lo que se extrajo el valor colocando su posición en los corchetes del arreglo *Numpy* de valores LST.

```
>>> valor = data[F][C]
```

Para la obtención de los datos en 24 horas se generó un archivo TXT por estación, en cual se registran cada hora los datos retornados del buscador por estación, eliminado el primer valor y añadiendo el nuevo al final. Los archivos son nombrados por el mismo valor del identificador de descarga de la estación. El proceso cada vez que es ejecutado obtiene los valores contenidos en este archivo y obtiene las gráficas.

Los GeoTIFF temporales son reproyectados al sistema de referencia con código EPSG:4326 y recortados en un cuadrante aproximado de 5 km durante el proceso de búsqueda del valor en la coordenada. Los valores del GeoTIFF son extraídos nuevamente y colocados en arreglos *Numpy* para su posterior mapeo.

### **Procesamiento de imágenes RBG en TC de Sentinel.**

Las imágenes Sentinel-2 son obtenidas con el script *sentineDyn*, el cual se encuentra en el **anexo de scripts**. Esta función realiza una petición WMS y escribe los datos en imagen PNG.

Tiene como parámetro el ingreso de la coordenada de la estación obtenida previamente, con la cual obtiene las

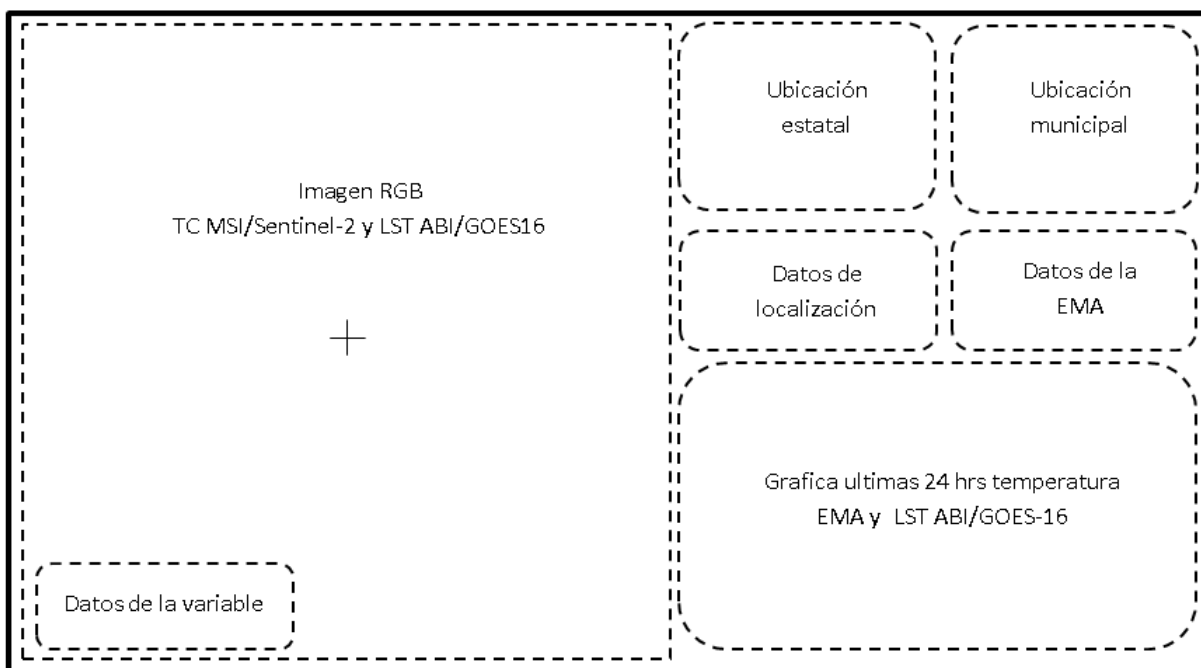
coordenadas de un cuadrante aproximado de 5 km. Las coordenadas son reproyectadas al sistema de referencia EPSG:4326 y colocadas en el parámetro BBOX de la petición WMS, obteniendo la última imagen Sentinel-2 por estación.

La imagen fue colocada como función de mapa base a los datos sobrepuestos de LST de GOES-16.

### Integración.

La integración de los datos se realizó con la generación una imagen PNG, en el cual están contenidas la gráfica y la imagen RGB.

El patrón de distribución de los elementos de salida en el PNG se encuentra en la Figura 4.12:



**Figura 4.12:** Patrón de distribución del PNG de integración final del proceso 1. Estaciones meteorológicas EMAS, GOES-16 y Sentinel-2.

### Automatización.

El proceso es generado cada hora para las 32 estaciones EMAS. En cada proceso es copiado el último NetCDF4 de los servidores y una vez finalizado el proceso es eliminado.

La automatización es realizada con una sentencia CRON<sup>15</sup> en Linux, el cual se encuentra en el **anexo de Scripts**.

<sup>15</sup>Es un demonio (proceso en segundo plano) que se ejecuta desde el mismo instante en el que arranca el sistema.

Diagrama de flujo.

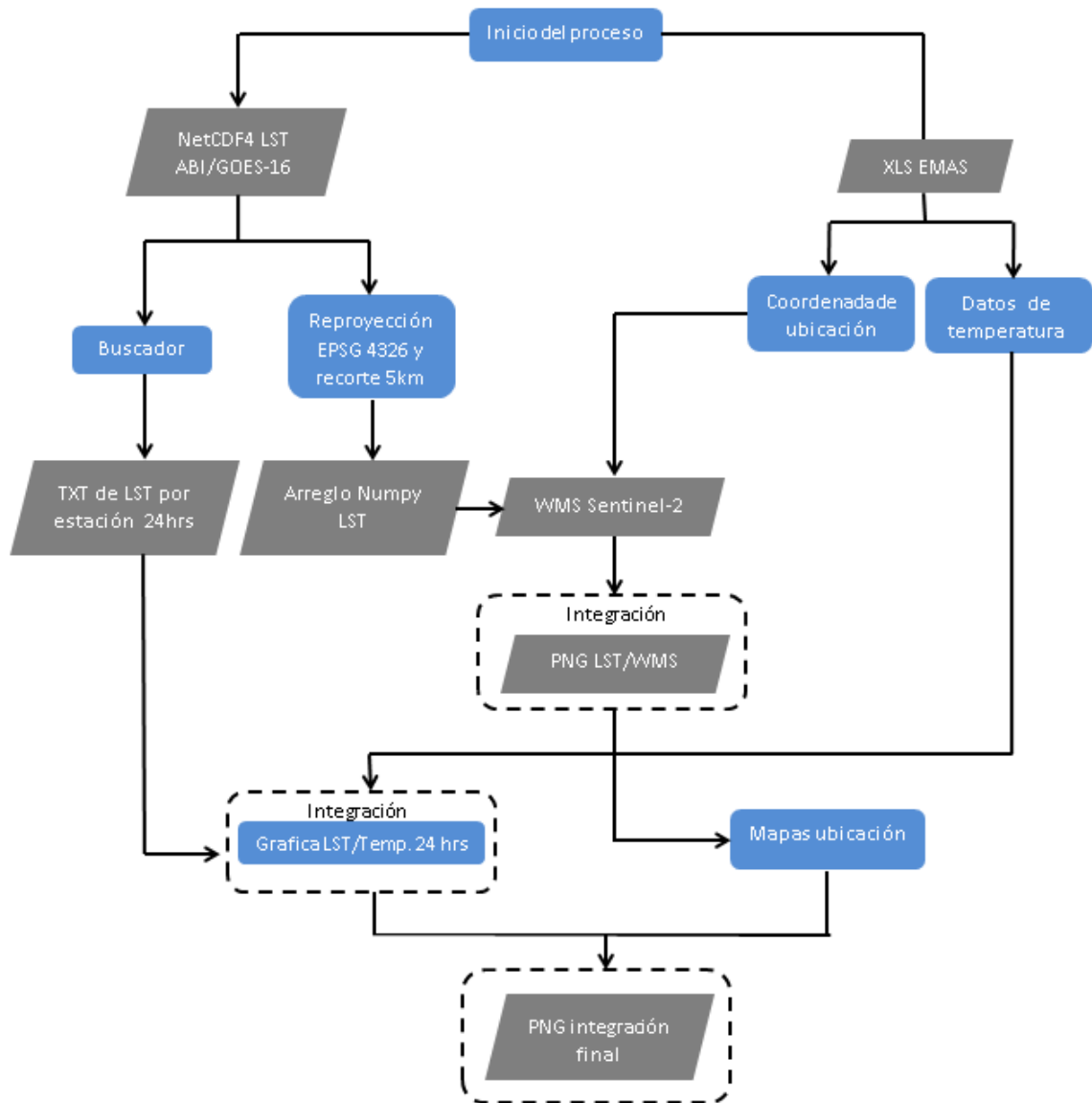


Figura 4.13: Diagrama de flujo del proceso 1. Estaciones meteorológicas EMAS, GOES-16 y Sentinel-2.



#### 4.6.2. Proceso 2. Boyas meteorológicas NOAA, Suomi-NPP y Sentinel-2

El proceso integra los datos del producto SST del sensor VIIRS/Suomi-NPP con los datos de temperatura superficial del mar de boyas meteorológicas de la NOAA, añadiendo el compuesto RGB en TC del último paso de Sentinel-2 en la ubicación de la boya con los datos sobrepuestos de SST del sensor VIIRS/Suomi-NPP.

##### Datos de Entrada.

- GeoTIFF del producto SST del sensor VIIRS/Suomi-NPP.
- TXT con registros de las últimas 24 hrs de la boya meteorológica de la NOAA.
- WMS del compuesto RGB en TC del último paso de Sentinel-2.

##### Datos de salida.

Gráficas con los datos de SST de VIIRS y temperatura superficial del mar de las últimas 24 horas en la ubicación de la boya meteorológica.

Compuesto RGB en TC de Sentinel-2 con los datos sobrepuestos de SST de VIIRS en un cuadrante de 5 km aproximadamente.

##### Procesamiento de datos boyas meteorológicas NOAA.

Los datos de las boyas meteorológicas en tiempo real son obtenidos del servidor National Data Buoy Center de la NOAA. Ver Figura 4.14

El url de conexión a los datos de descarga es el siguiente.

<https://www.ndbc.noaa.gov/data/realtime2/42055.txt>

El nombre del archivo TXT de descarga está asociado al identificador de la boya meteorológica. Se ocuparon 4 boyas, las cuales son las más cercanas al territorio mexicano. El identificador de las boyas fue almacenado en una variable de tipo lista.

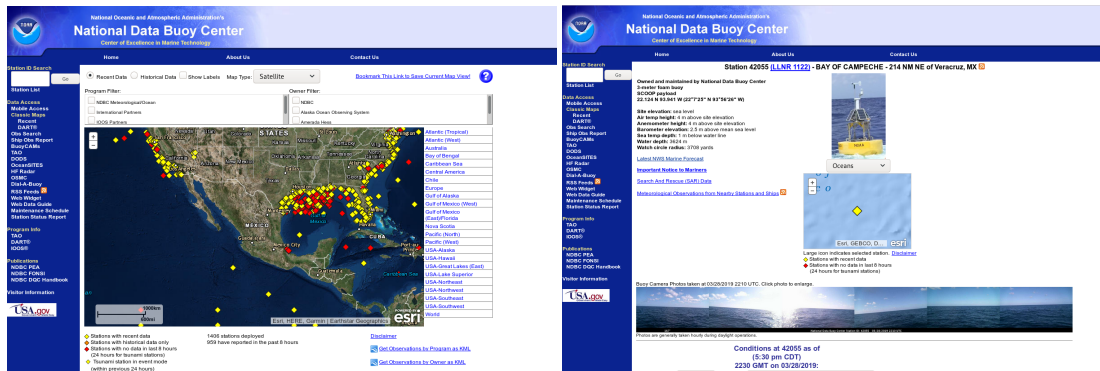


Figura 4.14: Página de la National Data Buoy Center NOAA con el mapa de boyas meteorológicas, tomadas de <https://www.ndbc.noaa.gov/>.

El método de recorrido y descarga de los datos fue similar al de las estaciones EMAS.

Los datos fueron descargados en el intervalo de 5 días, al ser el formato más accesible.

Se ocupó el módulo *Pandas* para convertir el TXT a CSV y crear un Dataframe. Se extrajeron los datos de temperatura superficial oceánica a partir del nombre de las columnas de Dataframe, almacenando solo el registro de las últimas 24 hrs.

La extracción de la coordenada de ubicación de la boya se realizó por un método de la técnica de Web Scraping<sup>16</sup> de la página HTML de la siguiente dirección.

[https://www.ndbc.noaa.gov/station\\_realtime.php?station=42055](https://www.ndbc.noaa.gov/station_realtime.php?station=42055)

El parámetro *station*, está relacionado con la clave de la boya. Se tuvo que realizar este proceso debido a que el archivo TXT no se encontraba las coordenadas de ubicación. Se ocuparon los módulos *Requests* y *BeautifulSoup* para realizar este proceso, ver Figura 4.15.

**Station 42055 (LLNR 1122) - BAY OF CAMPECHE - 214 NM NE of Veracruz, MX**

**Owned and maintained by National Data Buoy Center  
22.124 N 93.941 W (22°7'25" N 93°56'26" W)**

Figura 4.15: Etiqueta de la página donde se extrajeron datos de la boya mediante Web Scraping.

### Procesamiento de datos SST de VIIRS/Suomi-NPP.

Para la extracción de los datos del producto SST de VIIRS en las coordenadas de las boyas meteorológicas de la NOAA, se utilizó el mismo script denominado *buscador*, que fue utilizado en las estaciones EMAS, siguiendo el

<sup>16</sup>Técnica utilizada mediante programas de software para extraer información de sitios web.

mismo proceso de recorte, reproyección al sistema de referencia EPSG:4326 y búsqueda del valor en la coordenada.

Para la obtención de los datos de SST en 24 horas se generó el mismo procedimiento que el de las estaciones EMAS, al igual que la obtención de las gráficas.

### Procesamiento de imágenes RBG en TC de Sentinel-2.

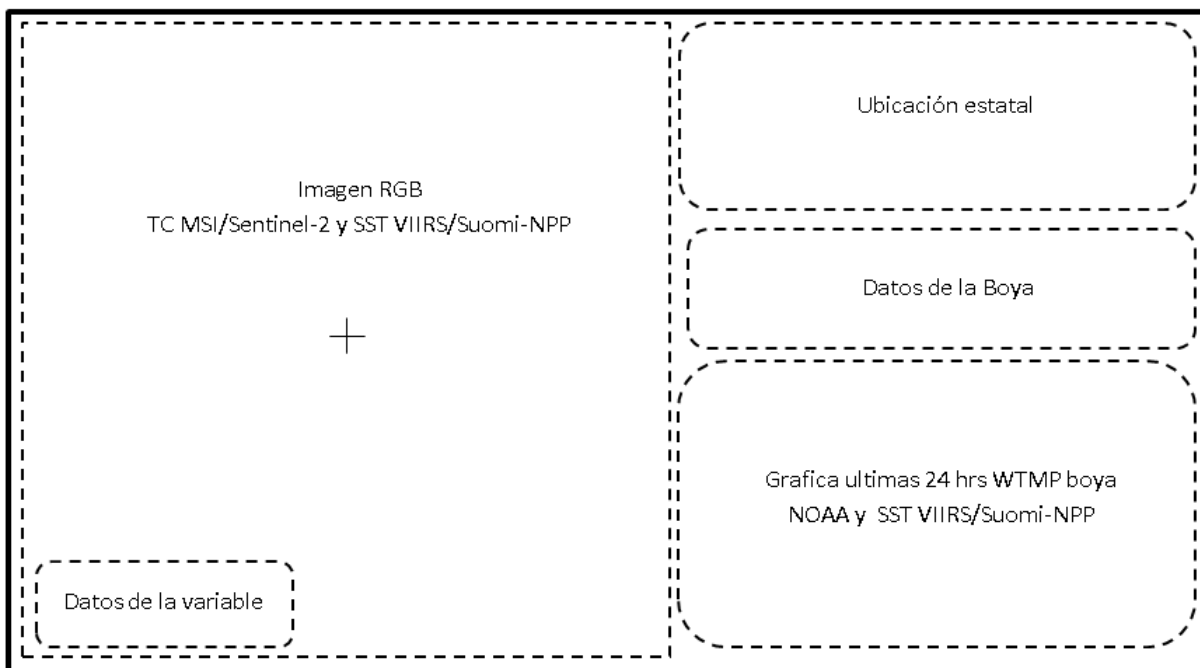
El proceso de obtención de la imagen Sentinel-2 fue con el mismo script *sentinelDyn* ocupado para las estaciones EMAS.

La imagen fue colocada como función de mapa base a los datos sobrepuestos de SST de VIIRS.

### Integración.

La integración de los datos se realizó con la generación una imagen PNG, en el cual están contenidas la gráfica y la imagen RGB.

El patrón de distribución de los elementos de salida en el PNG se encuentra en la Figura 4.16:



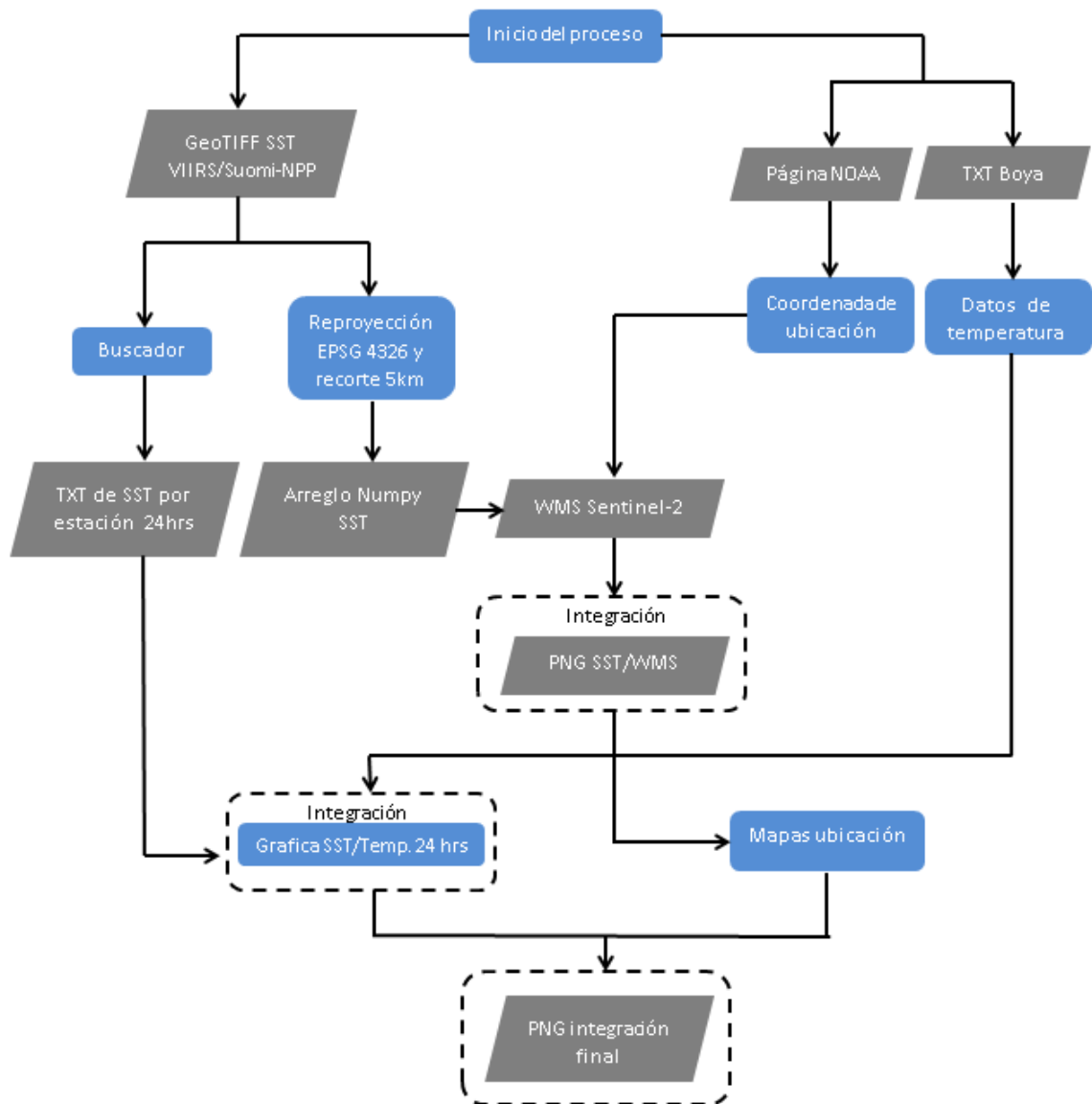
**Figura 4.16:** Patrón de distribución del PNG de integración final del proceso 2. Boyas meteorológicas NOAA, Suomi-NPP y Sentinel-2.

### Automatización.

El proceso es generado cada día a las 12:00 hrs para las 4 boyas meteorológicas de la NOAA. En cada proceso es copiado el último GeoTIFF de los servidores, una vez finalizado el proceso es eliminado.

La automatización es realizada con una sentencia CRON en Linux, la cual se encuentra en el **anexo de Scripts**.

**Diagrama de flujo.**



**Figura 4.17:** Diagrama de flujo del proceso 2. Boyas meteorológicas NOAA, Suomi-NPP y Sentinel-2.

### 4.6.3. Proceso 3. Puntos de calor, GOES-16, Suomi-NPP y Sentinel-2

El proceso integra los compuestos de TC, FT y SWIR, a través de imágenes RGB en coordenadas obtenidas del algoritmo de puntos de calor de GOES-16. El algoritmo de puntos de calor utiliza las características de las bandas de infrarrojo y visible para la caracterización de un punto caliente, el cual sirve para determinar la posición de posibles incendios.

Las regiones de las imágenes RGB son creadas a diferentes escalas de acuerdo a cada satélite.

En la Tabla 4.12 se muestra por satélite el compuesto generado y la extensión del cuadrante usado en el proceso 3.

**Tabla 4.12:** Compuestos generados por satélite y la extensión del cuadrante en el proceso 3.

Satélite	Compuesto RGB			Cuadrante aprox.
	TC	FT	SWIR	
GOES-16	*	*	-	100 km
Suomi-NPP	*	*	-	50 km
Sentinel-2	*	-	*	10 km

#### Datos de Entrada.

- NetCDF4 de las bandas 1,2,3 de radiancia del nivel L1b de GOES-16.
- NetCDF4 de las bandas 5,6,7 del producto CMI del nivel L2 de GOES-16
- GeoTIFF de las bandas 1,2,3,5,6,7 de radiancias de nivel L1 de VIIRS
- CSV de coordenadas de puntos de calor de GOES-16.
- WMS del compuesto RGB en TC del último paso de Sentinel-2.
- WMS del compuesto RGB de SWIR del último paso de Sentinel-2.

#### Datos de salida.

PNG integrado con compuestos RGB en TC de Goes-16, Suomi-NPP y Sentinel-2; compuesto RGB de FT de GOES-16 y Suomi-NPP; compuesto RGB de SWIR de Sentinel-2.

#### Procesamiento de CSV de puntos de calor.

Los puntos de calor son obtenidos de un archivo en formato CSV. Los valores de latitud, longitud y tiempo de paso satelital, son extraídos mediante el modulo *Pandas* y su función de lectura de datos CSV, la cual los convierte a Datasets facilitando la extracción por medio del nombre de la columna de datos.

Los puntos de calor son calculados para toda la región de Full Disk de GOES-16, por lo que los valores de las coordenadas fueron limitadas al cuadrante del territorio continental de México, mediante un condicional de máximo y mínimo.

Se generó una lista con los valores de las coordenadas en cada paso de GOES-16, procesando las imágenes para cada coordenada.

### **Procesamiento de composiciones RGB.**

#### **True Color.**

La obtención del RGB en TC de GOES-16 requirió los archivos NetCDF4 de las bandas 1, 2 y 3 del producto CMI. Con lo cual se obtuvo el verde sintético de acuerdo a la Formula 1.8.

La obtención del RGB en TC de Suomi-NPP ocupó los GeoTIFF de las bandas 1, 2 y 3 del producto radiancias de VIIRS.

El recorte en base a la coordenada fue realizada por el script *dataRegion*, el cual se encuentra en el anexo de scripts. Esta función genera un cuadrante respecto al valor de la coordenada, extrayendo los valores de las bandas.

El proceso de extracción, georreferenciación, reproyección y mapeo fue similar a la metodología descrita anteriormente para GOES-16 y Suomi-NPP. El script *dataTrueColor* realiza estos pasos, el cual se encuentra en el **anexo de scripts**.

La obtención del RGB de Sentinel-2 se realizó modificando el parámetro *LAYERS* de la petición WMS, colocando el valor 1-NATURAL-COLOR.

#### **Fire Temperature.**

La obtención del RGB de FT de GOES-16 ocupó los NetCDF4 de las bandas 5, 6 y 7 del producto CMI.

La obtención del RGB en TC de Suomi-NPP ocupó los GeoTIFF de las bandas 5, 6 y 7 del producto radiancias de VIIRS.

Se ocupó también el script *dataRegion* y la metodología de extracción, georreferenciación, reproyección y mapeo de GOES-16 y Suomi-NPP. El script *dataFireTemperature* realiza estos pasos, el cual se encuentra en el **anexo de scripts**.

#### **SWIR.**

La obtención del RGB de Sentinel, se realizó modificando el parámetro *LAYERS* de la petición WMS, colocando el valor 1-SWIR.

### Integración.

La integración de los datos se realizó con la generación una imagen PNG, en el cual están contenidas las imágenes de los compuestos generados. Se gráfico además la coordenada del punto de calor en cada imagen, colocando un identificador del punto, además de los tiempo de procesamiento del punto y el del paso del satélite.

El patrón de distribución de los elementos de salida en el PNG se encuentra en la Figura 4.18:

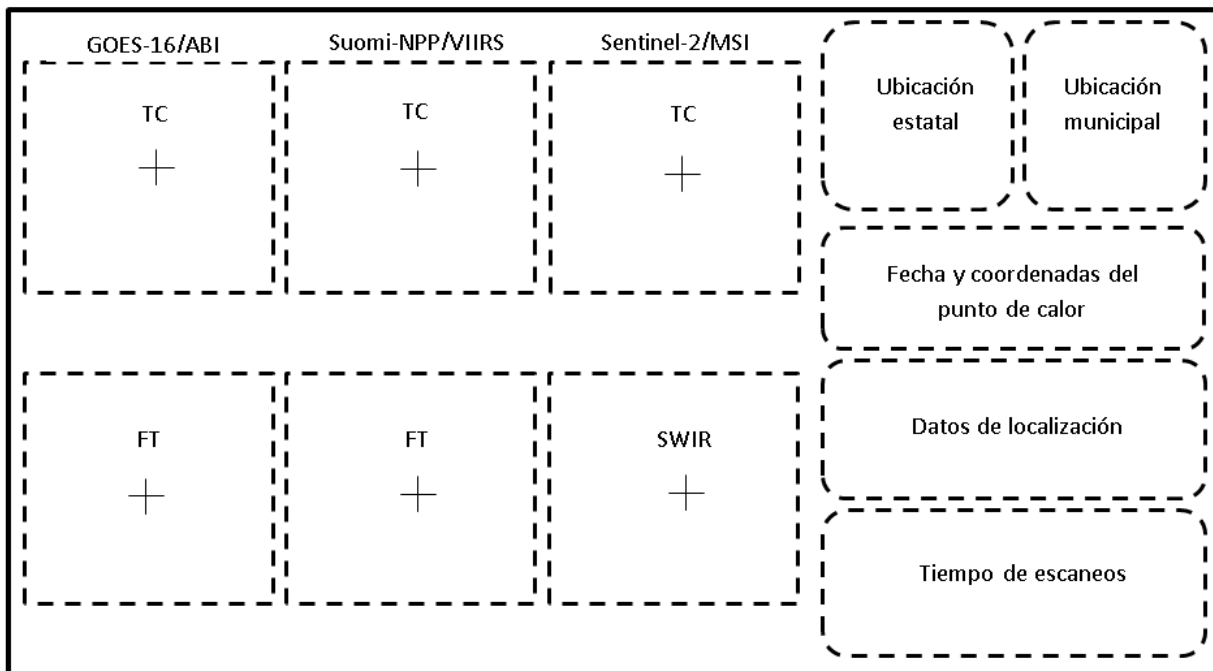


Figura 4.18: Patrón de distribución del PNG de integración final del proceso 3. Puntos de calor, GOES-16, Suomi-NPP y Sentinel-2.

### Automatización.

El proceso es generado cada hora de 10:00 a 18:00 hrs para todos los puntos en la región del territorio continental de México. En cada proceso es copiado el último NetCDF4, CSV y GeoTIIF de los servidores. Una vez finalizado el proceso son eliminados.

La automatización es realizada con un CRON en Linux el cual se encuentra en el **anexo de Scripts**.

Diagrama de flujo.

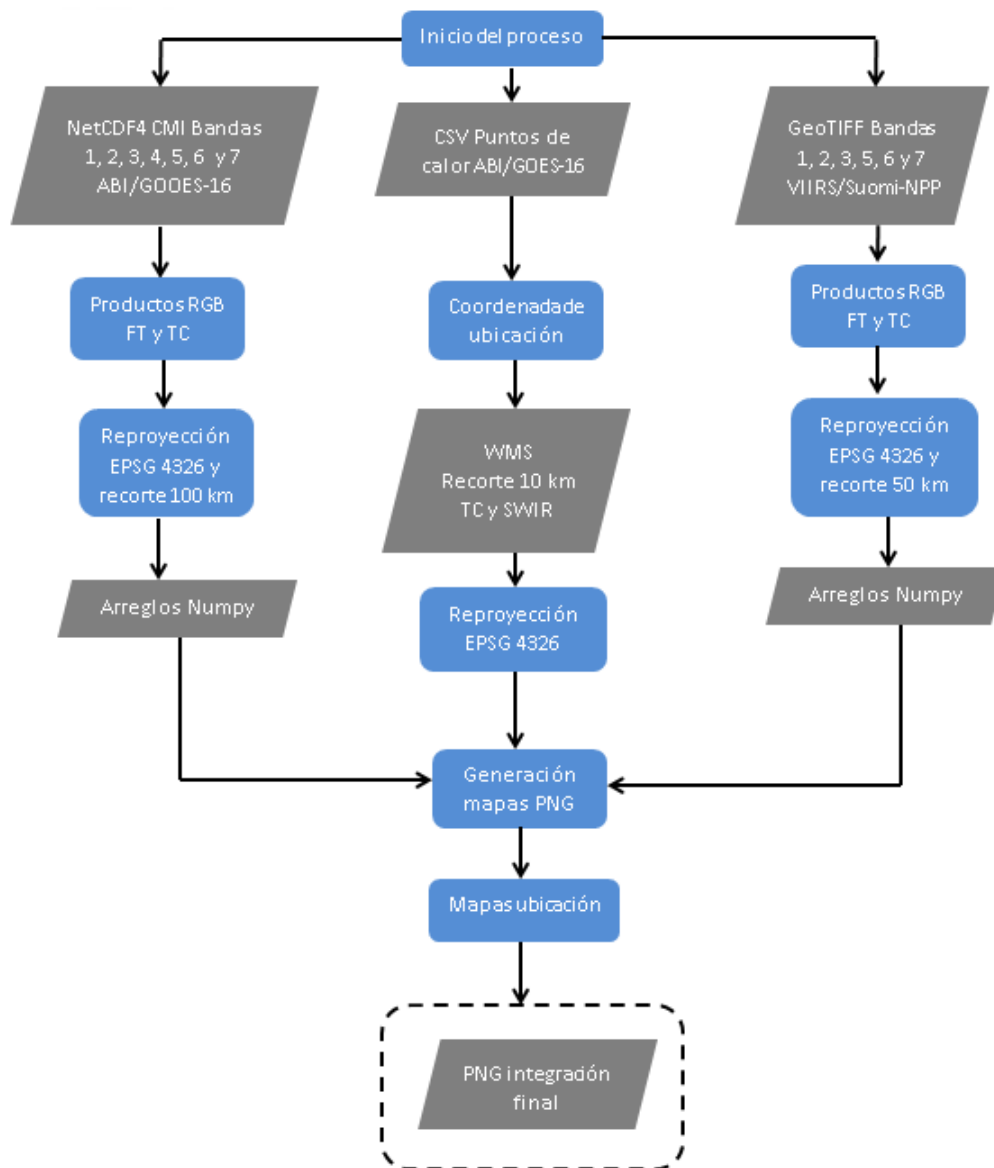


Figura 4.19: Diagrama de flujo del proceso 3. Puntos de calor, GOES-16, Suomi-NPP y Sentinel-2.





## Capítulo 5

# Resultados

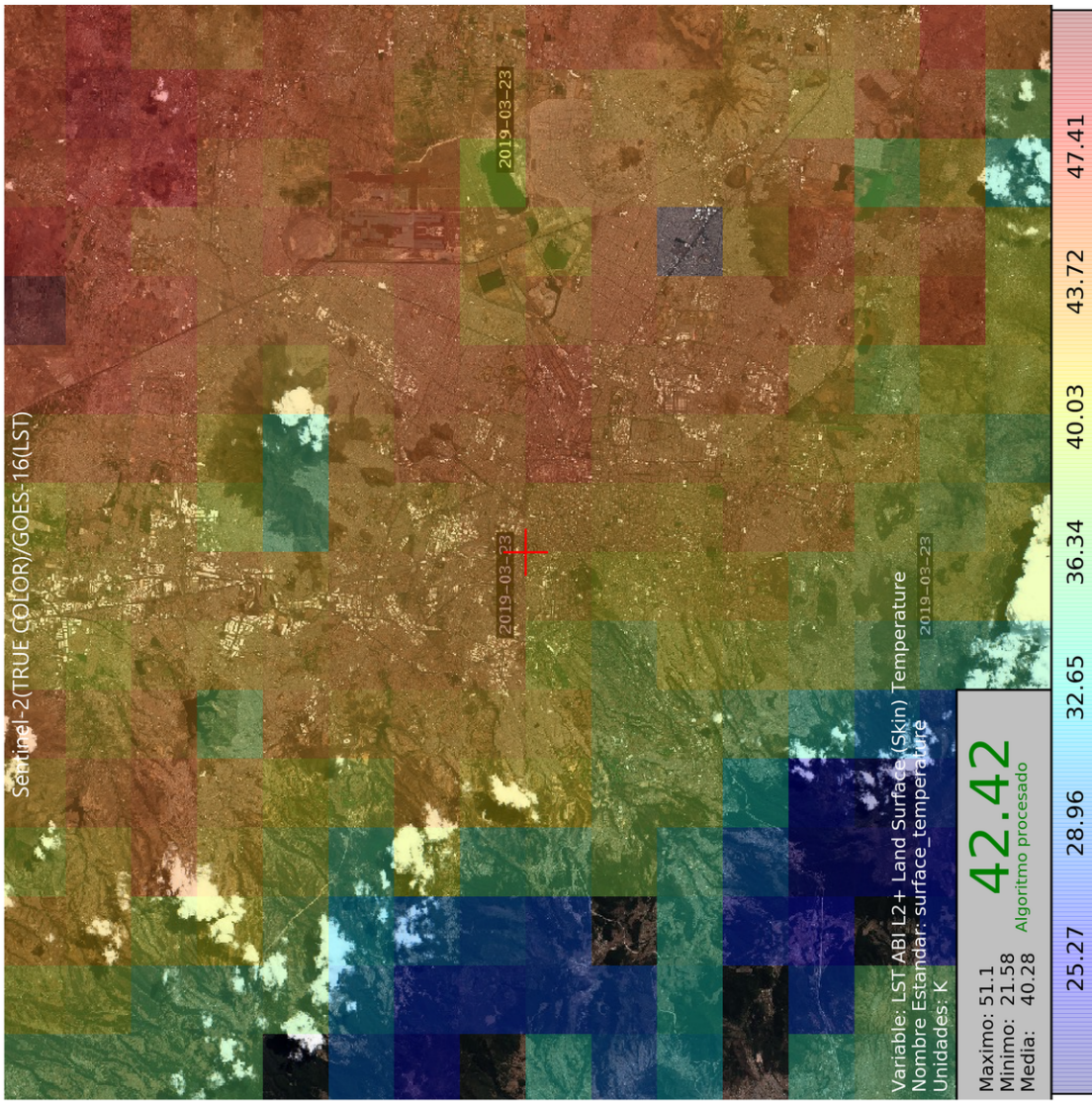
Se presentan las imágenes de integración generadas en cada proceso.

Al ser procesos automáticos, se eligieron imágenes por cada proceso en las cuales se presentan de forma general los casos comunes.

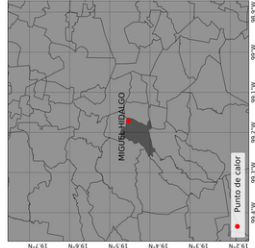
### 5.1. Proceso 1. Estaciones meteorológicas EMAS, GOES-16 y Sentinel-2

En el proceso 1, se eligieron dos imágenes, una imagen en la cual el valor de LST de ABI/GOES-16 se procesó en la ubicación de la estación EMA, ver Figura 5.1 y otra imagen en la que no, ver Figura 5.2.

Se presenta adicionalmente un mapa que muestra los valores del error medio cuadrático de las 32 estaciones EMAS, comparando la relación entre la temperatura del suelo y el aire a las 18:00 horas GMT. Los valores ocupados se generaron de forma automática en un archivo CSV, durante un mes, , ver Figura ??.



UBICACION



Sentinel-2 / MSI

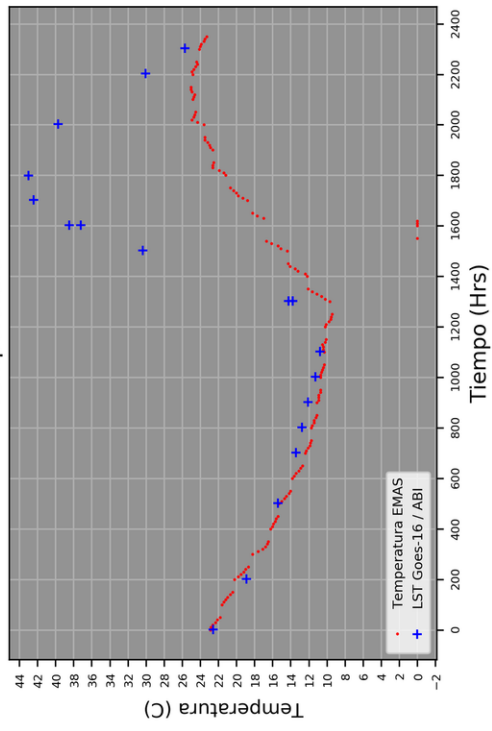
LOCALIZACION

Municipio: Miguel Hidalgo  
 Entidad: Ciudad de México  
 Clave Municipio: 016  
 Clave Entidad: 09  
 Clave Geoestadística: 09016

DATOS EMAS

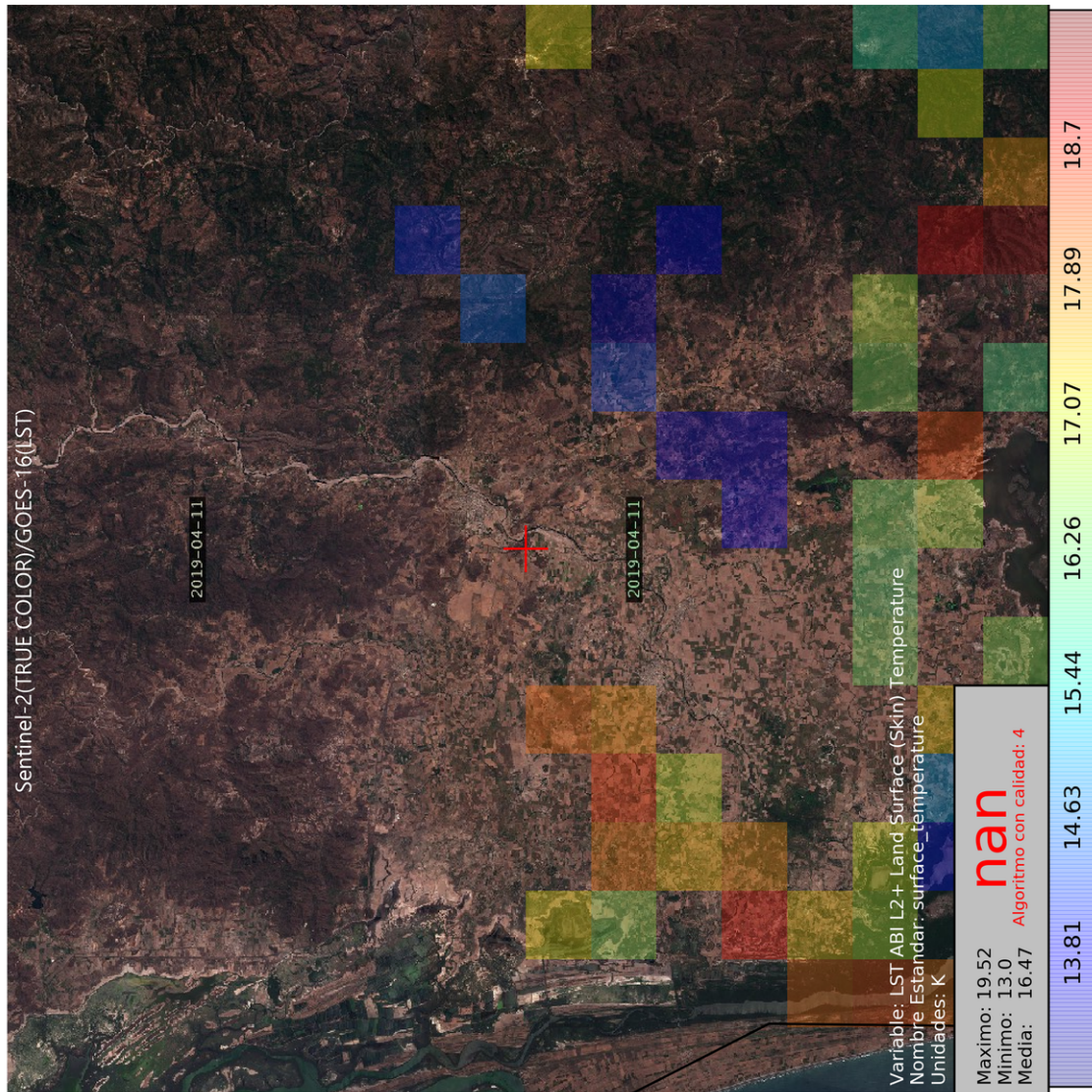
Estacion: C-BIOLOG\_IPN  
 Ubicacion: SMN EMAS  
 Operador: 2075  
 Altitud: 99.17 N  
 Longitud: 19.45 W

Temperatura

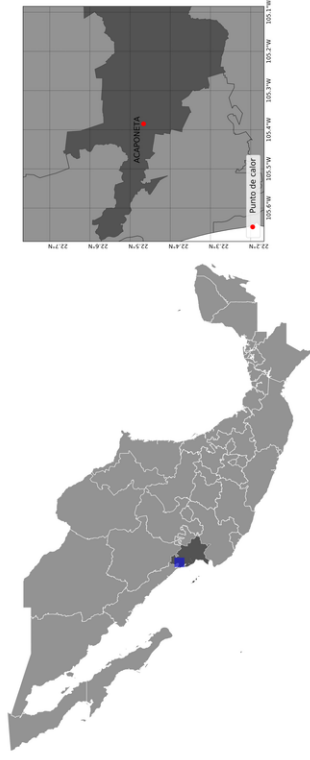


SULTADOS

Figura 5.1: Resultado de proceso 1, con LST de ABI/GOES-16 procesado en la ubicación de la estación EMA. Imagen generada el 25 de Marzo del 2019 a las 15:30 GMT.



## UBICACION



Sentinel-2 / MSI

**LOCALIZACION**  
 Municipio: Acaponeta  
 Entidad: Nayarit  
 Clave Municipio: 001  
 Clave Entidad: 18  
 Clave Geostatística: 18001

**DATOS EMAS**  
 Estación: ACAPONETA  
 Ubicación: ACAPONETA.1  
 Operador: SMN EMAS  
 Altitud: 29  
 Longitud: 105.39 N  
 Latitud: 22.47 W

## Temperatura

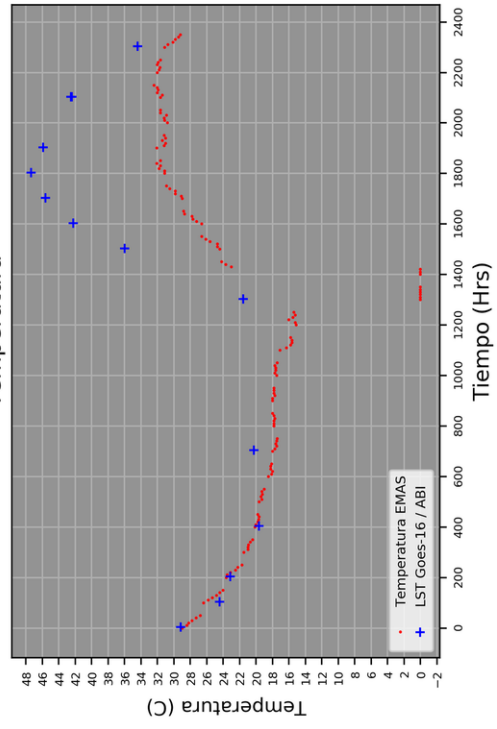


Figura 52: Resultado de proceso 1, con LST de ABI/GOES-16 no procesado en la ubicación de la estación EMA. Imagen generada el 12 de Abril del 2019 a las 13:00 GMT.

EMC de LST de ABI/GOES-16 respecto a temperatura del aire EMAS a las 18:00 GMT

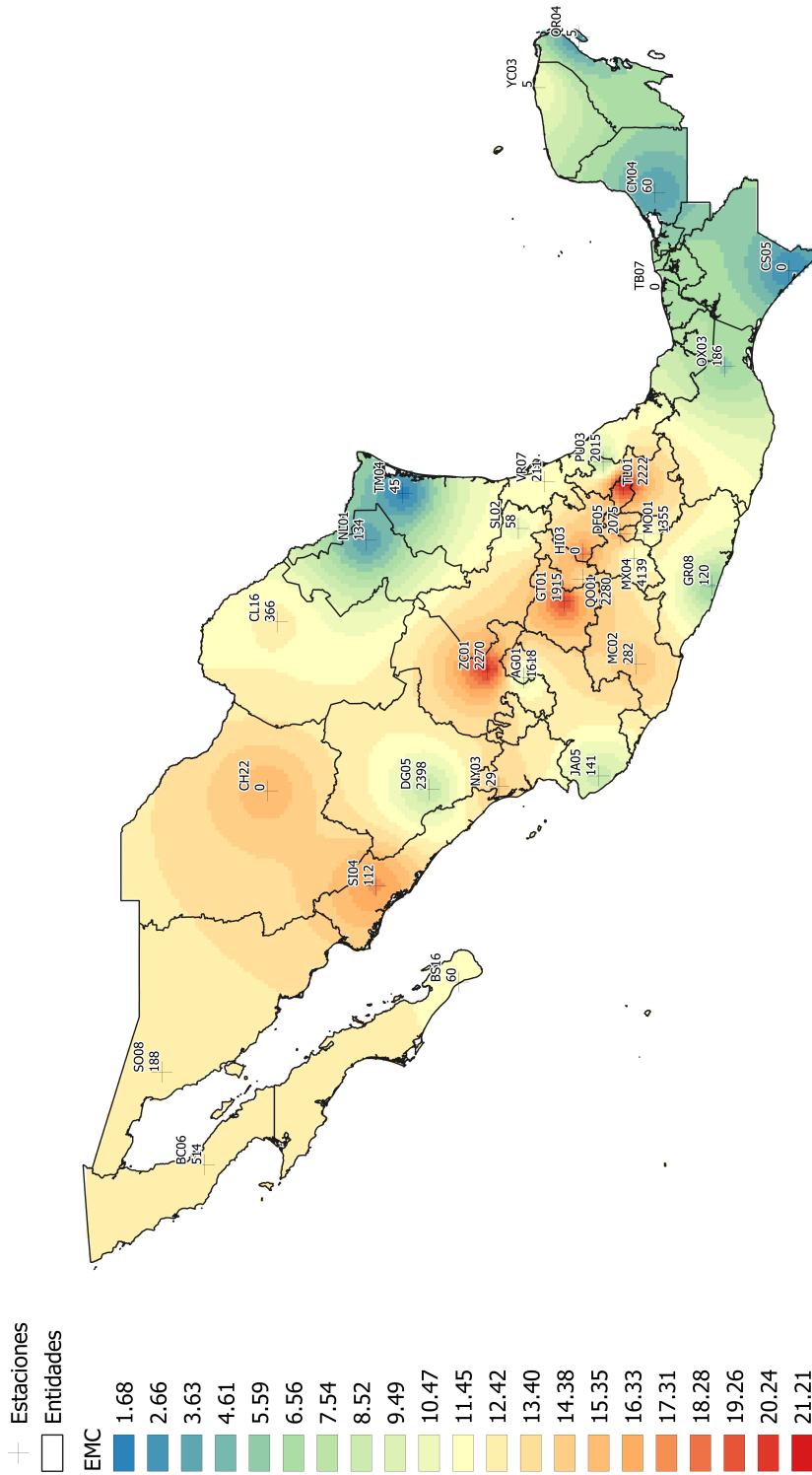


Figura 5.3: EMC de LST ABI/GOES-16 respecto a temperatura del aire de EMAS a las 18:00 GMT.

## **5.2. Proceso 2. Boyas meteorológicas NOAA, Suomi-NPP y Sentinel-2**

En el proceso 2, se eligieron dos imágenes, una imagen en la cual el valor de SST de VIIRS/Suomi-NPP se procesó en la ubicación de la boya meteorológica, , ver Figura 5.4 y otra imagen en la que no, , ver Figura 5.5.

UBICACION

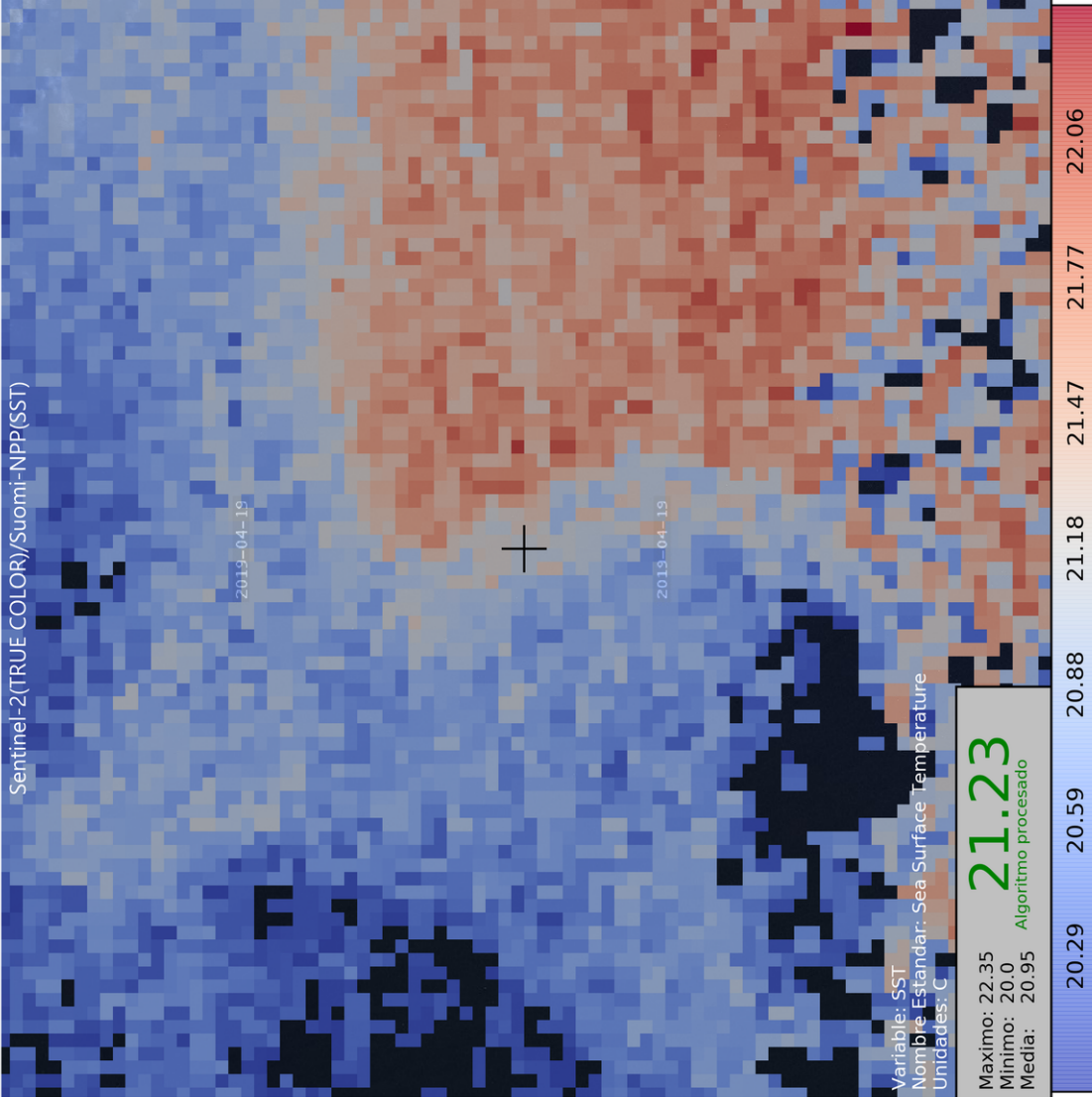
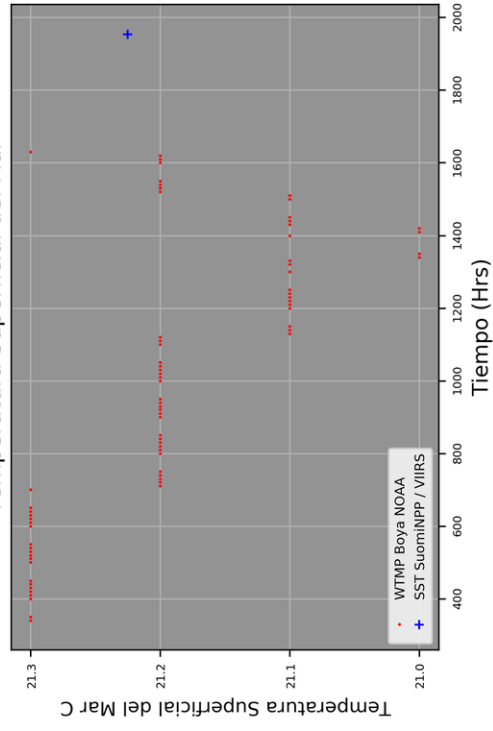


Sentinel-2 / MSI

DATOS BOYA NOAA

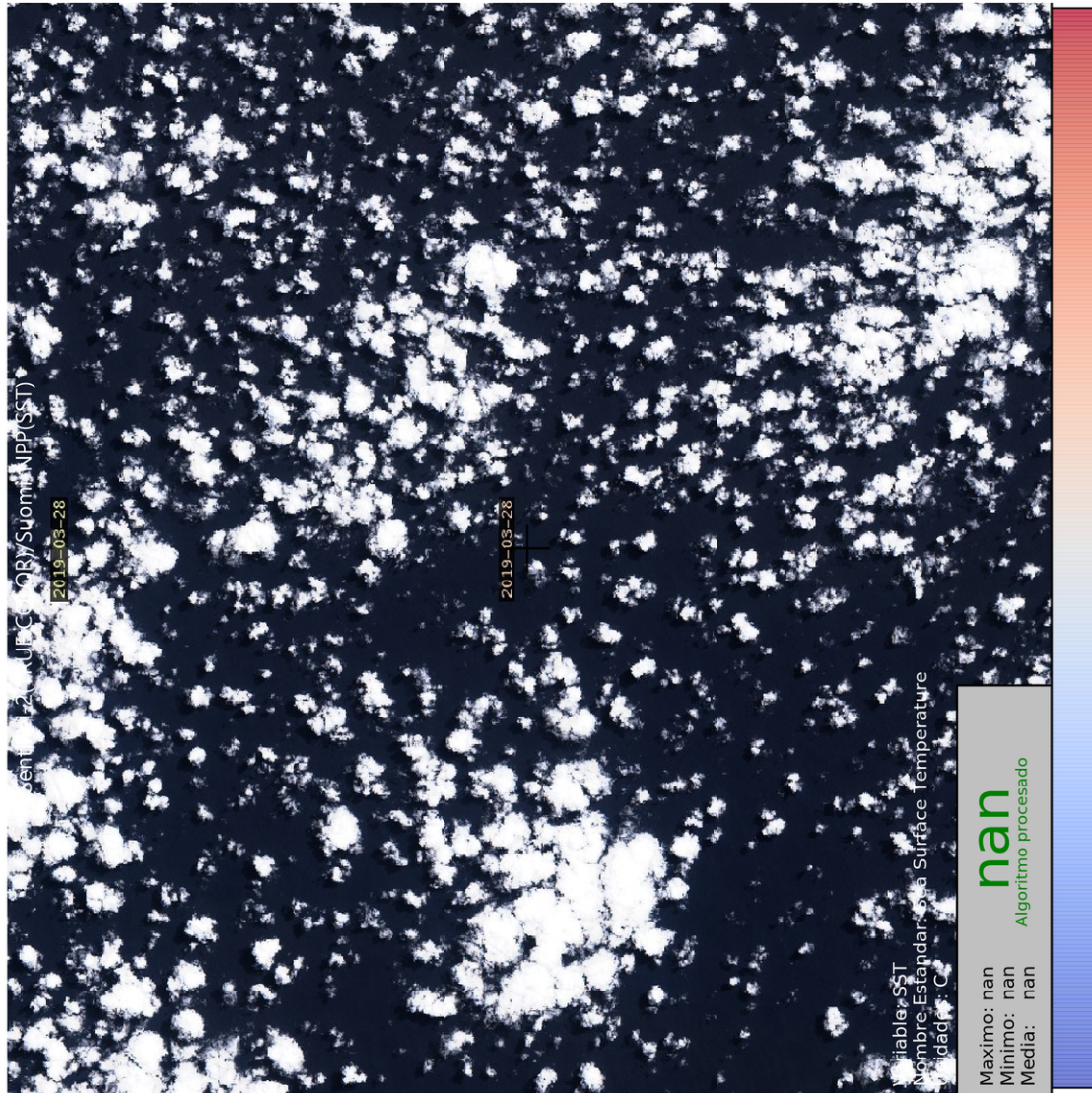
Boya: 42019  
 Ubicación: FREEPORT, TX - 60 NM South of Freeport, TX  
 Profundidad: m  
 Longitud: 95.35 W  
 Latitud: 27.91 N

Temperatura Superficial del Mar



RESULTADOS

Figura 5.4: Resultado de proceso 2, con SST de VIIRS/Suomi-NPP procesado en la ubicación de la boya meteorológica. Imagen generada el 22 de Abril del 2019 a las 12:00 GMT.



UBICACION



Sentinel-2 / MSI

DATOS BOYA NOAA

Boya: 42003  
 Ubicación: East GULF - 208 NM West of Naples, FL  
 Profundidad: el  
 Longitud: 85.61 W  
 Latitud: 25.93 N

Temperatura Superficial del Mar

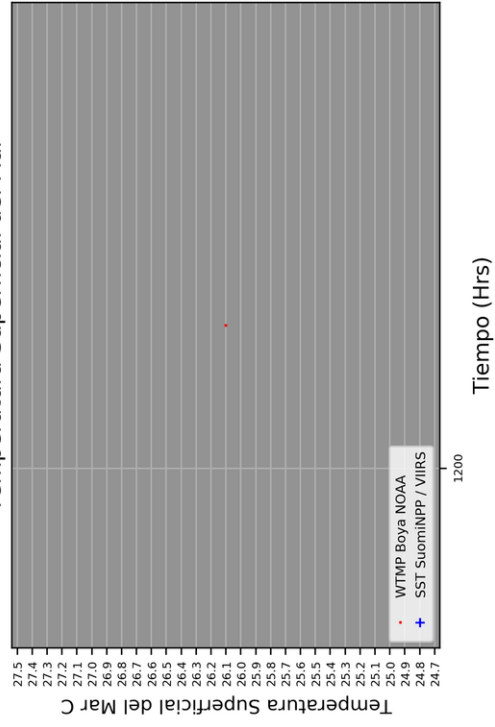


Figura 5.5: Resultado de proceso 2, con SST de VIIRS/Suomi-NPP no procesado en la ubicación de la baya meteorológica. Imagen generada el 30 de Marzo del 2019 a las 12:00 GMT.

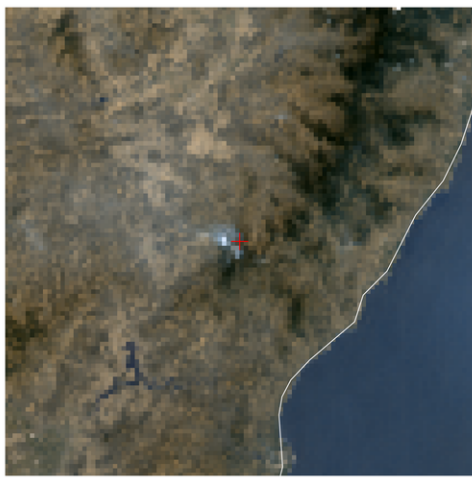


### **5.3. Proceso 3. Puntos de calor, GOES-16, Suomi-NPP y Sentinel-2**

En el proceso 3 se eligieron cuatro imágenes en la cual se detectó un punto de calor.

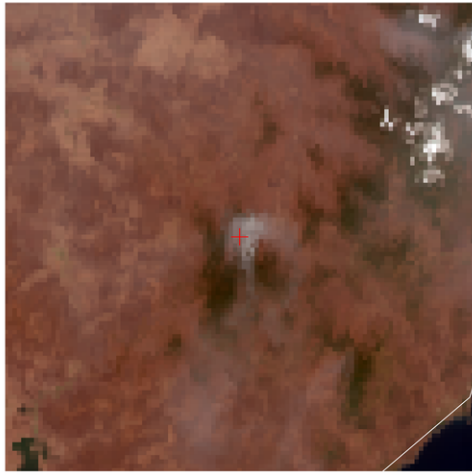
En la primera imagen los colores en los compuestos RGB muestran un incendio activo en los tres satélites (ABI/GOES-16, VIIRS/Suomi-NPP y MSI/Sentinel-2), ver Figura 5.6, en la segunda imagen en dos satélites (ABI/GOES-16 y VIIRS/Suomi-NPP), ver Figura 5.7, en la tercera imagen en un solo satélite (ABI/GOES-16), ver Figura 5.8 y en la cuarta imagen el punto de calor no muestra un incendio activo en ningún satélite, ver Figura ??.

GOES-16/Sensor ABI



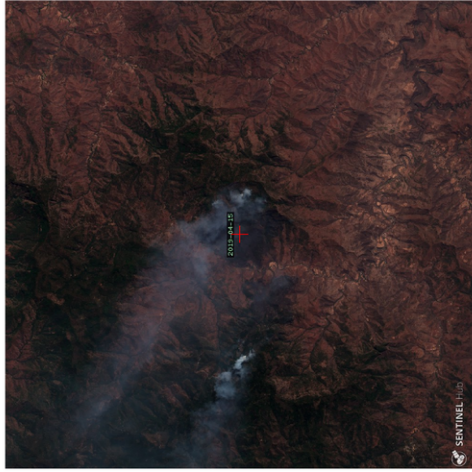
True Color [C01(0.47 um),C02(0.64 um),C03(0.86 um)]

Suomi-NPP/Sensor VIIRS



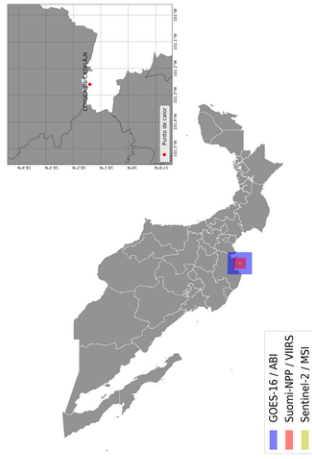
True Color [C03(0.48 um),C04(0.55 um),C05(0.67 um)]

Sentinel-2/Sensor MSI



True Color [C02(0.49 um),C03(0.55 um),C04(0.66 um)]

UBICACION



Punto de Calor GOES-16 detectado  
2019-01-07 12:30 GMT

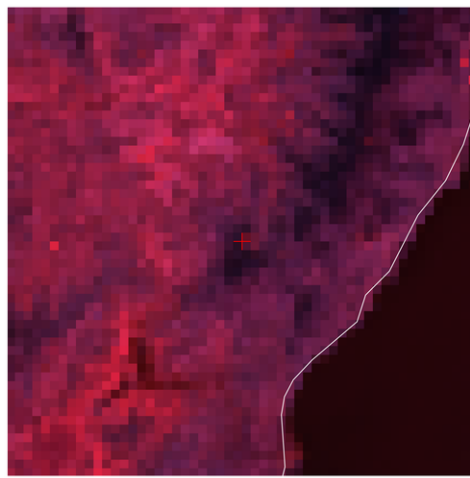
101.26 W 18.14 N

LOCALIZACION

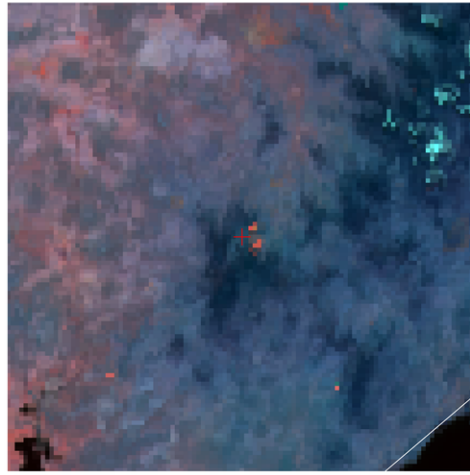
Municipio: Coyuca de Catalan  
Entidad: Guerrero  
Clave Municipio: 022  
Clave Entidad: 12  
Clave Geoestadística: 12022

TIEMPO DE ESCANEADO

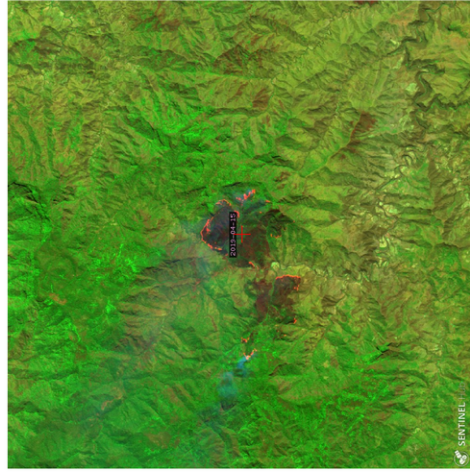
GOES-16/ABI: 2019/04/16 20:03 GMT  
Suomi-NPP/VIIRS: 2019/04/15 18:35 GMT  
Sentinel-2/MSI



Fire Temperature [C05(1.6 um),C06(2.25 um),C07(3.9 um)]



Fire Temperature [C10(1.61 um),C11(2.25 um),C12(3.7 um)]



SWIR [C03(0.55 um),C8A(0.83 um),C12(2.2 um)]

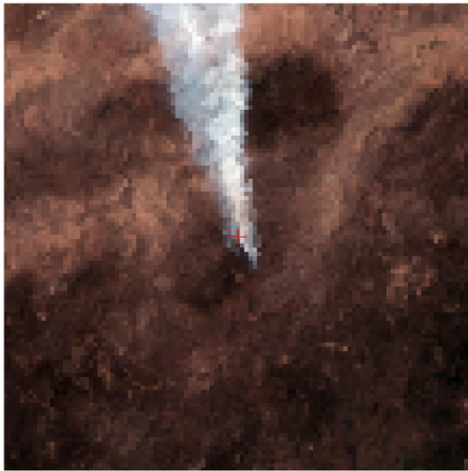
Figura 5.6: Resultado de proceso 3, con incendio activo observable en tres satélites (ABI/GOES-16, VIIRS/Suomi-NPP y MSI/Sentinel-2). Imagen generada el 15 de Abril del 2019 a las 18:35 GMT.

GOES-16/Sensor ABI



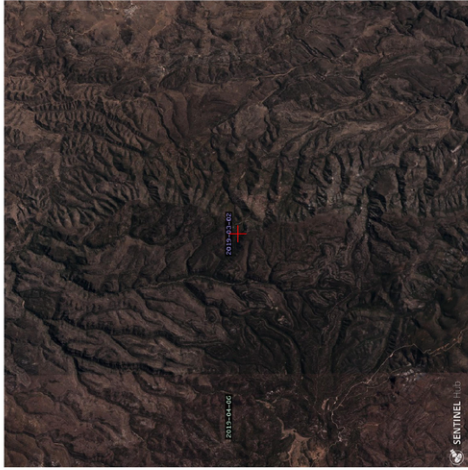
True Color [C01(0.47 um),C02(0.64 um),C03(0.86 um)]

Suomi-NPP/Sensor VIIRS



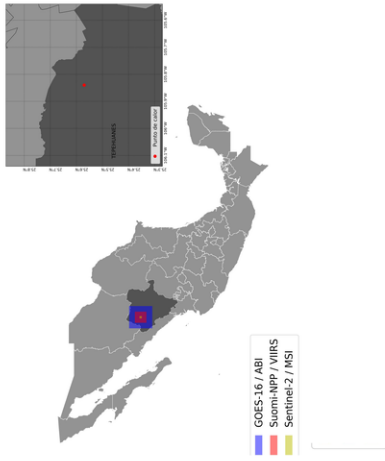
True Color [C03(0.48 um),C04(0.55 um),C05(0.67 um)]

Sentinel-2/Sensor MSI



True Color [C02(0.49 um),C03(0.55 um),C04(0.66 um)]

UBICACION



Punto de Calor GOES-16 detectado  
2019-04-17 18:28:08 GMT

105.84 W 25.57 N

LOCALIZACION

Municipio:

Entidad:

Clave Municipio:

Clave Entidad:

Clave Geostadística:

Tepehuanes  
Durango  
035  
10  
10035

TIEMPO DE ESCANEADO

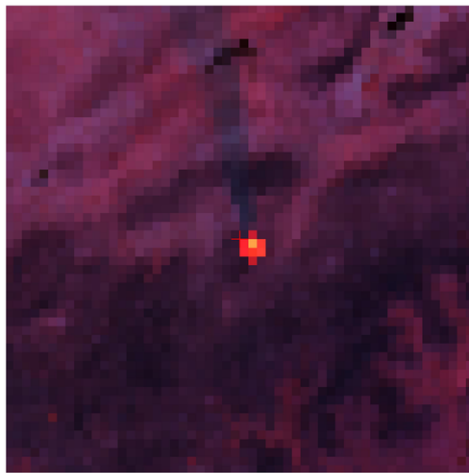
GOES-16/ABI:

Suomi-NPP/VIIRS

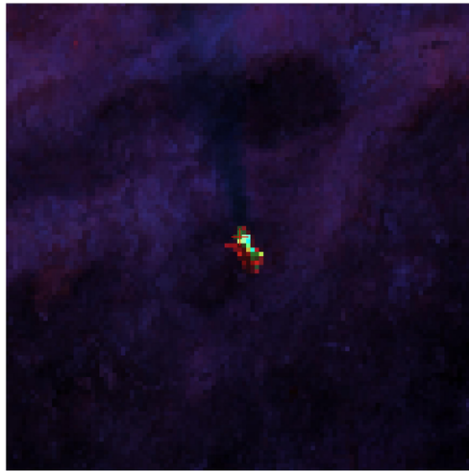
Sentinel-2/MSI

2019/04/17 20:05 GMT  
2019/04/17 19:36 GMT

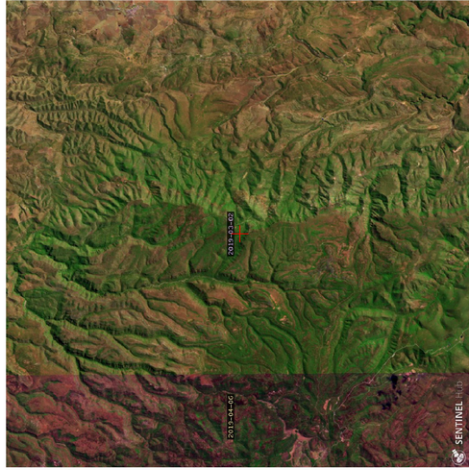
Fire Temperature [C05(1.6 um),C06(2.25 um),C07(3.9 um)]



Fire Temperature [C10(1.61 um),C11(2.25 um),C12(3.7 um)]



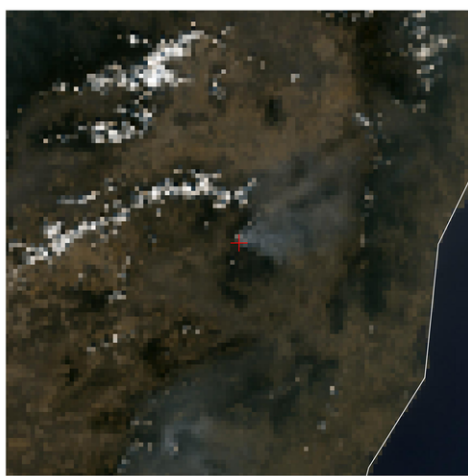
SWIR [C03(0.55 um),C8A(0.83 um),C12(2.2 um)]



5. RESULTADOS

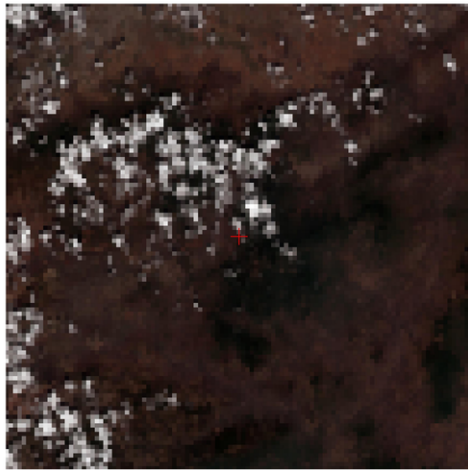
Figura 5.7: Resultado de proceso 3, con incendio activo observable en dos satélites (ABI/GOES-16 y VIIRS/Suomi-NPP). Imagen generada el 17 de Abril del 2019 a las 19:36 GMT.

GOES-16/Sensor ABI



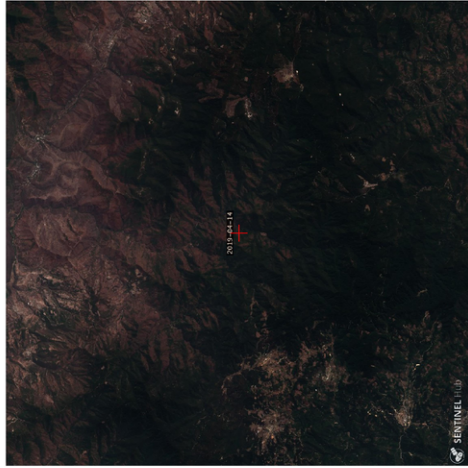
True Color [C01(0.47 um),C02(0.64 um),C03(0.86 um)]

Suomi-NPP/Sensor VIIRS



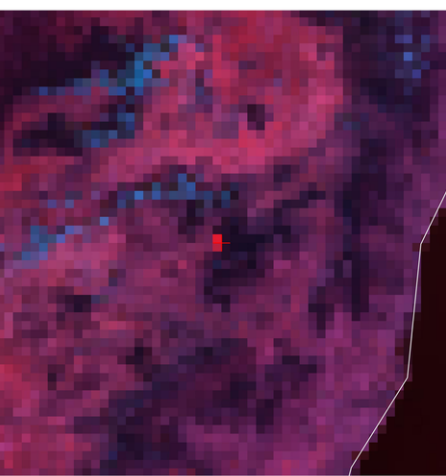
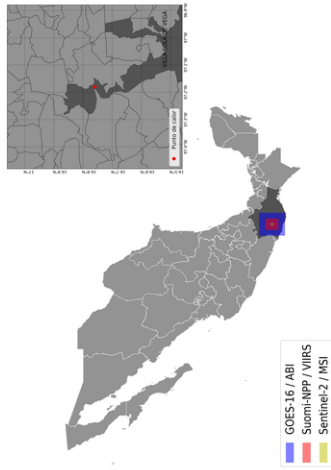
True Color [C03(0.48 um),C04(0.55 um),C05(0.67 um)]

Sentinel-2/Sensor MSI

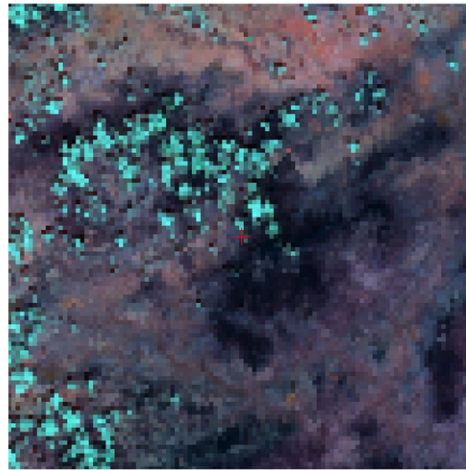


True Color [C02(0.49 um),C03(0.55 um),C04(0.66 um)]

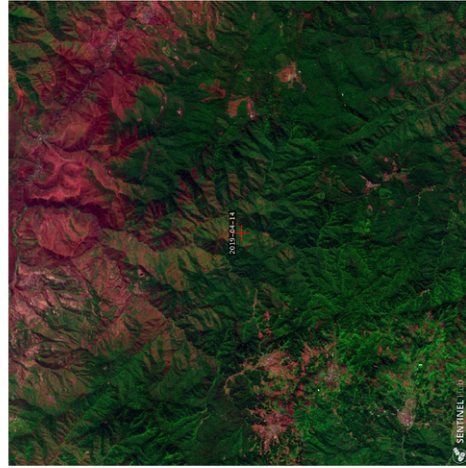
UBICACION



Fire Temperature [C05(1.6 um),C06(2.25 um),C07(3.9 um)]



Fire Temperature [C10(1.61 um),C11(2.25 um),C12(3.7 um)]



SWIR [C03(0.55 um),C8A(0.83 um),C12(2.2 um)]

Punto de Calor GOES-16 detectado  
2019-04-18 16:27:17 GMT  
**97.18 W 16.76 N**

LOCALIZACION

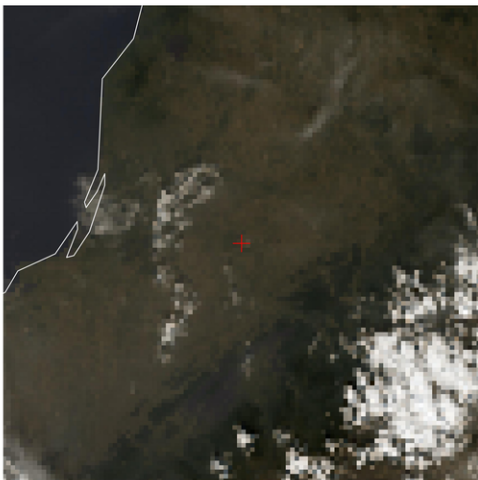
Municipio: Villa Sola de Vega  
Entidad: Oaxaca  
Clave Municipio: 277  
Clave Entidad: 20  
Clave Geostatística: 20277

TIEMPO DE ESCANEADO

GOES-16/ABI: 2019/04/18 18:05 GMT  
Suomi-NPP/VIIRS: 2019/04/17 19:36 GMT  
Sentinel-2/MSI

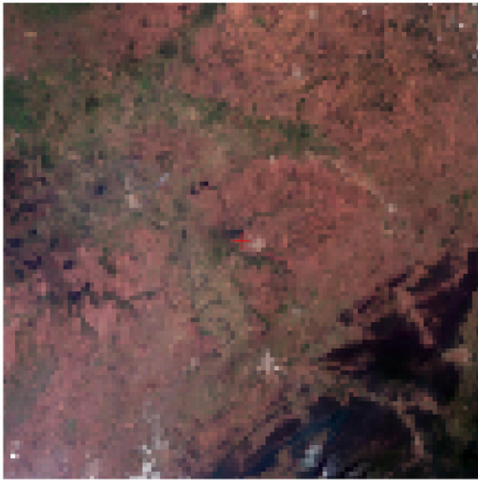
Figura 5.8: Resultado de proceso 3, con incendio activo observable en un satélite (ABI/GOES-16). Imagen generada el 17 de Abril del 2019 a las 19:36 GMT.

GOES-16/Sensor ABI



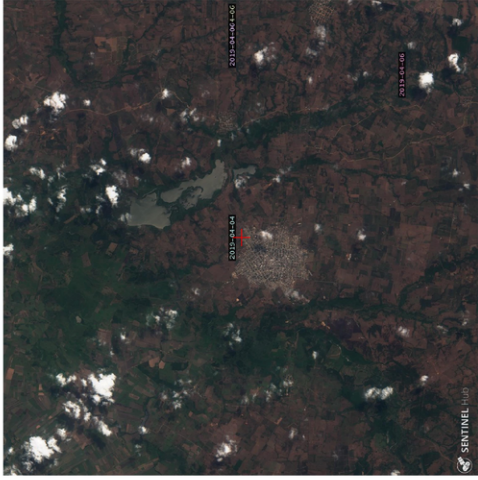
True Color [C01(0.47 um),C02(0.64 um),C03(0.86 um)]

Suomi-NPP/Sensor VIIRS



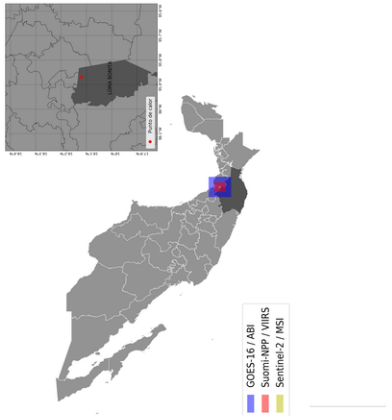
True Color [C03(0.48 um),C04(0.55 um),C05(0.67 um)]

Sentinel-2/Sensor MSI



True Color [C02(0.49 um),C03(0.55 um),C04(0.66 um)]

UBICACION



Punto de Calor GOES-16 detectado  
2019-04-07 17:37:26 GMT

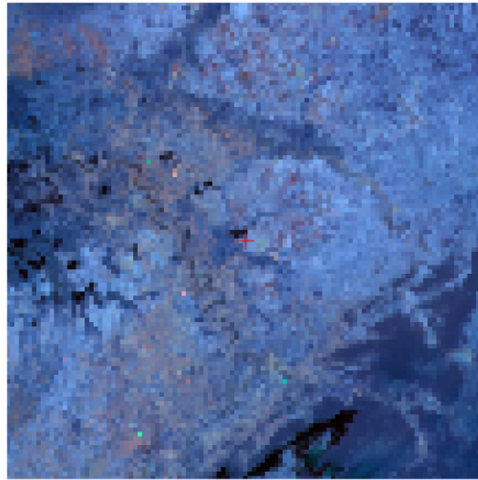
**95.87 W 18.12 N**

LOCALIZACION

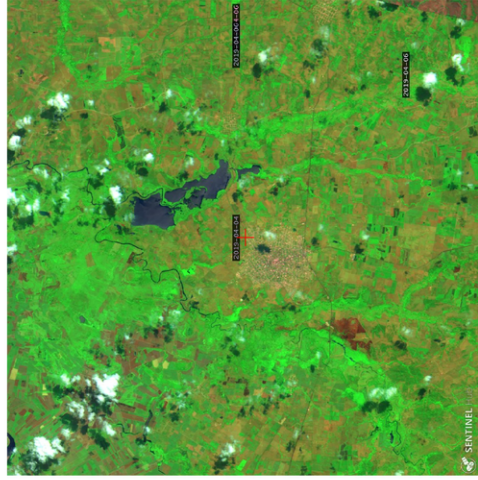
Municipio: Loma Bonita  
Entidad: Oaxaca  
Clave Municipio: 044  
Clave Entidad: 20  
Clave Geoestadística: 20044

TIEMPO DE ESCANEADO

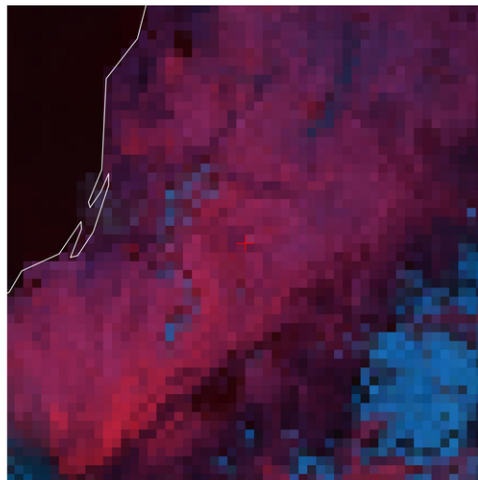
GOES-16/ABI: 2019/04/07 20:05 GMT  
Suomi-NPP/VIIRS: 2019/04/06 19:42 GMT  
Sentinel-2/MSI: .



Fire Temperature [C10(1.61 um),C11(2.25 um),C12(3.7 um)]



SWIR [C03(0.55 um),C8A(0.83 um),C12(2.2 um)]



Fire Temperature [C05(1.6 um),C06(2.25 um),C07(3.9 um)]

**Figura 5-9:** Resultado de proceso 3, con ningún incendio activo observable. Imagen generada el 17 de Abril del 2019 a las 19:36 GMT.

## Capítulo 6

# Conclusiones

La utilización de módulos y bibliotecas para el manejo de datos espaciales disponibles en el lenguaje de programación Python en la metodología desarrollada, permitió la integración de datos provenientes de satélites de órbita polar y geostacionaria de forma óptima con scripts reducidos. Esto a pesar de presentar diferentes características en cuanto al formato de origen y a las resoluciones de los datos. Fue necesario el estudio previo de las características del satélite y de sus datos, además del procesamiento de sus productos, ya que los procesos desarrollados con la metodología, aprovechaban las cualidades de los datos de cada satélite para analizar un fenómeno.

En el proceso 1 (estaciones meteorológicas EMAS, GOES-16 y Sentinel-2), las gráficas muestran un acercamiento de los valores del producto LST de ABI/GOES-16 con el valor de temperatura del aire de las EMAS. Estos dos valores siguen la misma tendencia a lo largo de las 00:00 a las 14:00 GMT en la mayoría de las estaciones. El mapa del error medio cuadrático a las 18:00 horas GMT, muestra las estaciones que menos error tuvieron en la hora con mas diferencia entre las dos variables. Las estaciones con más error, están ubicadas en zonas urbanas identificadas por la imagen Sentinel-2 y presentan mayor altitud.

En el proceso 2 (boyas meteorológicas NOAA, Suomi-NPP y Sentinel-2), las gráficas muestran un acercamiento de los valores del producto SST de VIIRS/Suomi-NPP con el valor de temperatura superficial del mar de las boyas meteorológicas de la NOAA. El único valor de SST que se procesa al día varia de 0.1 a 0.2 grados de diferencia con respecto al valor de la boya.

En el proceso 3 (puntos de calor, GOES-16, Suomi-NPP y Sentinel-2), la integración de composiciones RGB en una sola imagen, facilita la discriminación de puntos de calor que están asociados a incendios, ya que en incendios activos se puede observar la columna de humo en TC, la temperatura del pixel en FT y el área quemada e incendio activo con SWIR. También la resolución temporal de cada satélite permite dar un monitoreo continuo a un incendio activo.

En los tres procesos, las etapas de extracción, tratamiento, georreferencia, re proyección y mapeo, al realizarse de forma automática con sentencias CRON en Linux, facilitaron el análisis de los resultados. Los scripts de Python que realizaron estas etapas, trataron de desarrollarse de forma genérica, esto para que la mejora o el desarrollo de nuevos procesos fuera más eficiente.

# **Anexos**





# Anexos A

## Formatos

### A.1. NetCDF4

NetCDF (Network Common Data Form) es un conjunto de bibliotecas de software y formatos de datos autodescriptionales e independientes de la máquina, para datos científicos orientados a matrices. Fue desarrollado por el Centro de Programas de Unidata, con el objetivo de crear un formato de archivo que permitiera compartir datos atmosféricos. El formato fue y está diseñado para ser portátil, independiente de la plataforma, escalable y agregable.[LOC,2012]

NetCDF4, un formato mejorado de NetCDF3 introducido en 2008. NetCDF4 se basa en HDF5, con la cual introdujo una nueva estructura de agrupación y varias características para facilitar una mejor autodescription. Un uso de la estructura de grupo es simular directorios de archivos en una jerarquía.[LOC,2012] El modelo del dato consta de tres partes:

#### **Variables.**

Matrices n-dimensionales de datos. Las variables en los archivos NetCDF4 pueden ser de uno de los seis tipos (char, byte, short, int, float, double).

#### **Dimensiones.**

Describe los ejes de las matrices de datos. Una dimensión tiene un nombre y una longitud. Los archivos NetCDF4 pueden contener como máximo una dimensión ilimitada.

#### **Atributos.**

Anexan a las variables o archivos pequeñas notas o metadatos suplementarios. Los atributos son siempre valores

escalares o matrices 1D, que pueden asociarse a una variable o al archivo en su totalidad.

## A.2. GeoTIFF

TIFF (Tagged Image File Format) es un formato de archivo informativo para almacenar imágenes de mapa de bits. Es prevalente en la industria gráfica y en la fotografía profesional por su versatilidad y compresión no destructiva.[EARTHDATA,2019]

El formato de archivo de imagen etiquetado georreferenciado GeoTIFF, se usa como un formato de intercambio para imágenes raster georreferenciadas. GeoTIFF se usa ampliamente en los sistemas de datos espaciales de la NASA.

Los usuarios actuales y los proveedores de datos con formato GeoTIFF se basan principalmente en poder lograr una interoperabilidad basada en el uso del mismo código base de software (libgeotiff) y en las convenciones comúnmente conocidas.[EARTHDATA,2019]

## A.3. WMS

El servicio WMS (Web Mapping Service) definido por el OGC produce mapas de datos referenciados espacialmente, de forma dinámica a partir de información geográfica. Este estándar internacional define un "mapa" como una representación de la información geográfica en forma de un archivo de imagen digital conveniente para la exhibición en una pantalla de ordenador. Los mapas producidos por WMS se generan normalmente en un formato de imagen como PNG, GIF o JPEG.[Web Map Service,2019]

Las operaciones WMS pueden ser invocadas usando un navegador estándar realizando peticiones en la forma de URL. El contenido de tales URL depende de la operación solicitada. Concretamente, al solicitar un mapa, la URL indica qué información debe ser mostrada en el mapa, qué porción de la tierra debe dibujar, el sistema de coordenadas de referencia, la anchura y la altura de la imagen de salida.[Web Map Service,2019]

## Anexos B

# Scripts Bash

Scripts genéricos para copia de archivos, con reducidas modificaciones en cada proceso.

1. NetCDFGOES16.sh copia el ultimo archivo NetCDF4 del directorio de ABI.

```
server="lanot@cirrus"
dirIn="/data/output/abi/12/conus"
dirOut="/home/lanot/proceso/data_GOES16/C0"$1
NC=`ssh $server 'ls -t -l $dirIn '/*CMI*C0'$1 '*_*|_head_-1'`
echo $NC
scp $server:$NC $dirOut
```

2. GeoTiffSuomiNPP.sh copia los últimos archivos GeoTIFF del directorio de VIIRS.

```
server="lanot@cirrus"
dirIn="/data/output/viirs/11/global"
dirOut="/home/lanot/proceso/data_SuomiNPP/C"$1
GTIF1=`ssh $server 'ls -t -l $dirIn '/*.*19*ch'$1'*.SVM.*_*|_head_-1'`
GTIF2=`ssh $server 'ls -t -l $dirIn '/*.*18*ch'$1'*.SVM.*_*|_head_-1'`
GTIF3=`ssh $server 'ls -t -l $dirIn '/*.*20*ch'$1'*.SVM.*_*|_head_-1'`
scp $server:$GTIF1 $dirOut
scp $server:$GTIF2 $dirOut
scp $server:$GTIF3 $dirOut
```

3. CSVPuntosCalor.sh copia el ultimo archivo CSV del directorio de puntos de calor de GOES-16.

```
server="lanot@cirrus"
dirIn="/home/lanot/datos_terra2/puntos_goes"
dirOut="/home/lanot/proceso/data_PuntosCalor"
CSV=`ssh $server 'ls -t -l $dirIn '*_|_head_-1'`
scp $server:$dirIn/'$CSV $dirOut
```



# Anexos C

## Scripts Python

### C.1. Funciones genéricas

Funciones genéricas utilizadas en repetidas ocasiones en los scripts de procesos.

1. genericSat.py contiene las funciones genericas utilizadas en los procesos.

```
def creaTiff(data,proj4,xmin, ymin, xmax, ymax, nx, ny):
    xres,yres = (xmax - xmin) / float(ny),(ymax - ymin) / float(nx)
    geotransform = (xmin, xres, 0, ymax, 0, -yres)
    dst_ds = gdal.GetDriverByName('GTiff').Create('tmp.tif', ny, nx, 1, gdal.GDT_Float32)
    dst_ds.SetGeoTransform(geotransform)
    srs = osr.SpatialReference()
    srs.ImportFromProj4(proj4)
    dst_ds.SetProjection(srs.ExportToWkt())
    dst_ds.GetRasterBand(1).WriteArray(data)
    dst_ds.FlushCache()
    dst_ds = None

def coordenadasVentana(x,y,offset):
    urlon,lllon,urlat,lllat=x + offset,x - offset, y + offset,y - offset
    coorVentana = [lllon, urlat, urlon, lllat]
    return coorVentana

def dataVentana(x,y,offset):
    coorVentana = coordenadasVentana(x,y,offset)
    ds = gdal.Open('tmp.tif')
    gdal.Warp('tmp_4326.tif',ds,options=gdal.WarpOptions(dstSRS='EPSG:4326',dstNodata=-9999.000))
    ds = gdal.Open('tmp_4326.tif')
    gdal.Translate('tmp_4326_rec.tif',ds,options=gdal.TranslateOptions(projWin=coorVentana,noData=np.nan))
    ds = gdal.Open('tmp_4326_rec.tif')
    data = ds.ReadAsArray()
```

```

data[data == -9999.000] = np.nan
ds = None
os.remove('tmp.tif')
os.remove('tmp_4326.tif')
os.remove('tmp_4326_rec.tif')
return data

def rebin(a, shape):
    sh = shape[0], a.shape[0]//shape[0], shape[1], a.shape[1]//shape[1]
    return a.reshape(sh).mean(-1).mean(1)

def normaliza(data):
    data = (data - np.nanmin(data))*255/(np.nanmax(data)-np.nanmin(data))
    return data

def correcionGamma(data, gamma):
    data = np.power(data, gamma)
    return data

def compuestoRGB(r, g, b, entero):
    if entero == True:
        rgb = (np.dstack((r, g, b))).astype('uint8')
    else:
        rgb = (np.dstack((r, g, b)))
    return rgb

def buscadorTif(ds, x, y):
    geotransform = ds.GetGeoTransform()
    xMin, yMax = geotransform[0], geotransform[3]
    resx, resy = geotransform[1], geotransform[5]
    data = ds.ReadAsArray()
    yMin = yMax - resx*data.shape[0]
    xOffset, yOffset = int((x - xMin) / resx), int((y - yMin) / resy)
    value = data[yOffset, xOffset]
    return value

def calculaTicks(dataMin, dataMax):
    inter = 0
    if dataMin >= 0:
        inter = dataMax - dataMin
    elif dataMin < 0 and dataMax > 0:
        inter = dataMax + abs(dataMin)
    elif dataMin < 0 and dataMax <= 0:
        inter = abs(dataMin) - abs(dataMax)
    tick1 = dataMin + inter/8
    tick2 = dataMin + inter*2/8
    tick3 = dataMin + inter*3/8
    tick4 = dataMin + inter*4/8
    tick5 = dataMin + inter*5/8
    tick6 = dataMin + inter*6/8
    tick7 = dataMin + inter*7/8
    return tick1, tick2, tick3, tick4, tick5, tick6, tick7

```

```

def plotRGBbus(rgb, varBus, coorVentana, x, y, offset, salida, varName, stdName, unidades):
    salida = '/home/lanot/proceso/output/'+salida
    plt.figure(figsize=(10,10))
    ax = plt.axes(projection=crrs.PlateCarree())
    ax.coastlines(resolution='50m', color='k')
    ax.set_extent([coorVentana[0], coorVentana[2], coorVentana[3], coorVentana[1]])
    sen = plt.imread('tmp_4326Sen.tif')
    plt.imshow(sen, extent=[coorVentana[0], coorVentana[2], coorVentana[3], coorVentana[1]])
    plt.imshow(rgb, extent=[coorVentana[0], coorVentana[2], coorVentana[3], coorVentana[1]],
    alpha=0.7, cmap='coolwarm', vmin=20)

    tick1, tick2, tick3, tick4, tick5, tick6, tick7 = calculaTicks(20, np.nanmax(rgb))
    cb = plt.colorbar(ticks=[tick1, tick2, tick3, tick4, tick5, tick6, tick7], orientation
    = 'horizontal', pad = 0, shrink = 0.83)
    cb.ax.set_xticklabels([str(round(tick1, 2)), str(round(tick2, 2)), str(round(tick3, 2)),
    str(round(tick4, 2)), str(round(tick5, 2)), str(round(tick6, 2)), str(round(tick7, 2))])
    cb.outline.set_visible(True)
    cb.ax.tick_params(width = 0)
    cb.ax.xaxis.set_tick_params(pad=-20)
    cb.ax.tick_params(axis='x', colors='black', labelsz=10)

    plt.plot(x, y, 'k+', markersize = 20)
    currentAxis = plt.gca()
    currentAxis.add_patch(Rectangle((x-offset, y-offset/1.0), offset*0.75, offset*0.18,
    alpha=1, facecolor='silver', zorder = 3, ec="black", lw=1.0))
    plt.text(x-offset/1.02, y-offset/1.25, 'Variable: '+varName+' \nNombre_Estandar: '+
    stdName+' \nUnidades: '+unidades, fontsize=8, color = 'white')
    plt.text(x-offset/1.02, y-offset/1.035, 'Maximo: '+str(round(np.nanmax(rgb), 2))
    + '\nMinimo: '+str(round(np.nanmin(rgb), 2)) + '\nMedia: '+str(round(np.nanmean
    (rgb), 2)), fontsize=8)

    if varBus == np.nan:
        calColor = 'b'
        calidad = 'Algoritmo_no_procesado'
    else:
        calColor = 'g'
        calidad = 'Algoritmo_procesado'
    plt.text(x-offset/1.6, y-offset/1.09, str(round(float(varBus), 2)), fontsize=
    24, color=calColor)
    plt.text(x-offset/1.45, y-offset/1.05, calidad, fontsize=7, color=calColor)
    imagen = salida+str(x)+'_'+str(y)+'.png'
    plt.savefig(imagen, dpi=300, bbox_inches='tight', pad_inches=0)
    sen, rgb = None, None
    return imagen

def plotRGB(rgb, coorVentana, x, y, salida):
    salida = '/home/lanot/proceso/output/'+salida
    plt.figure(figsize=(10,10))
    ax = plt.axes(projection=crrs.PlateCarree())
    ax.coastlines(resolution='50m', color='w')
    ax.set_extent([coorVentana[0], coorVentana[2], coorVentana[3], coorVentana[1]])

```



```

plt.imshow(rgb, extent=[coorVentana[0], coorVentana[2], coorVentana[3], coorVentana[1]])
plt.plot(x, y, 'r+', markersize = 20)
imagen = salida+str(x)+'_'+str(y)+'.png'
plt.savefig(imagen, dpi=300, bbox_inches='tight', pad_inches=0)
rgb = None
return imagen

```

```

def creaGrafica(tickx, ticky, tiempo, tiempoG16, variable, satVariable, nomVariable,
unidad, simbolo, estacion, ubicacion):

```

```

    fig = plt.figure(figsize=(6,4))
    tick_spacing = tickx
    tick_spacing_y = ticky

    ax = plt.axes()
    ax.set_facecolor('#939393')
    ax.plot(tiempo, variable, simbolo, markersize=1, label='Temperatura_EMAS')
    ax.plot(tiempoG16, satVariable, 'b+', markersize=5, label='LST_Goes-16_/ABI')
    ax.legend(loc='lower_left', facecolor='white', fontsize=7)
    ax.yaxis.set_tick_params(labelsize=6, color='k', labelcolor='k')
    ax.xaxis.set_tick_params(labelsize=6, color='k', labelcolor='k')
    ax.xaxis.set_major_locator(ticker.MultipleLocator(tick_spacing))
    ax.yaxis.set_major_locator(ticker.MultipleLocator(tick_spacing_y))

    plt.title(nomVariable, color='k')
    plt.ylabel(nomVariable+'_'+unidad, color='k')
    plt.xlabel('Tiempo_(Hrs)', fontsize = 11, color='k')
    plt.grid(True)
    fig.savefig('grafica.png', dpi=300, bbox_inches='tight',
pad_inches=0.2, facecolor="white")

```

## C.2. Datos de entrada

Scripts genéricos para generar y eliminar los datos de entrada en los procesos, presentan reducidas modificaciones en cada proceso.

1. dataGOES16.py contiene funciones para la generación y eliminación de los datos de entrada de GOES-16.

```
import glob, os
from time import strptime, strftime

def copiaNetCDF(banda):
    os.system('sh_/home/lanot/proceso/NetCDFGOES16.sh_'+banda)

def revisaNetCDF(banda):
    files = glob.glob('/home/lanot/proceso/data_GOES16/C0'+banda+'/*.nc')
    fecha = files[0][files[0].find('_c')+2:files[0].find('.nc')-3]
    fechaG16 = strftime("%Y/%m/%d_%H:%M", strptime(fecha, "%Y%j%H%M%S"))
    return fechaG16

def borraNetCDF(banda):
    os.system('rm_/home/lanot/proceso/data_GOES16/C0'+banda+'/*')

def borraTodo():
    for i in range(7):
        if i+1 != 4:
            borraNetCDF(str(i+1))

def extraeNetCDF():
    for i in range(7):
        if i+1 != 4:
            copiaNetCDF(str(i+1))
```

2. dataSuomiNPP.py contiene funciones para la generación y eliminación de los datos de entrada de Suomi-NPP.

```

import glob, os
from time import strptime, strftime

def copiaGTif(banda):
    os.system('sh_/home/lanot/proceso/GeoTiffSuomiNPP.sh_' + banda)

def revisaGTif(banda, paso):
    files = glob.glob('/home/lanot/proceso/data_SuomiNPP/C'+banda+'/*.'
+pasos+'*.npp*.tif')
    fecha = files[0][files[0].find('/2019')+1:files[0].find('.npp')]
    fechaSNPP = strftime("%Y/%m/%d_%H:%M", strptime(fecha, "%Y.%m.%d.%H%M"))
    return fechaSNPP

def borraGTif(banda):
    os.system('rm_/home/lanot/proceso/data_SuomiNPP/C'+banda+'/*')

def borraTodo():
    for i in range(12):
        if 2 < i+1 < 6:
            borraGTif('0'+str(i+1))
        if 9 < i+1 < 13:
            borraGTif(str(i+1))

def extraeGTif():
    for i in range(12):
        if 2 < i+1 < 6:
            copiaGTif('0'+str(i+1))
        if 9 < i+1 < 13:
            copiaGTif(str(i+1))

```

3. dataSentinel2.py contiene funciones para la generación y eliminación de los datos de entrada de Sentinel-2.

```
import requests , os
import genericSat as gs
from pyproj import Proj , transform
from time import strftime , gmtime

def coorVentanaSentinelHub(x,y,offset):
    km = 111.12
    offset = offset*km*1000
    p1,p2 = Proj(init='epsg:4326'),Proj(init='epsg:3857')
    coorCent = transform(p1,p2,x,y)
    coorVentanaProj = gs.coordenadasVentana(coorCent[0],coorCent[1],offset)
    return coorVentanaProj

def borraJPG( RGB ):
    os.system( 'rm - /home/lanot/proceso/data.Sentinel2 /'+RGB+'/* ' )

def borraTodo():
    borraJPG( 'TC' )
    borraJPG( 'SWIR' )

def extraeWMS(x,y,offset,LAYERS,salida,path,RGB):
    coorVentana = coorVentanaSentinelHub(x,y,offset)
    BBOX = str(coorVentana[0])+','+str(coorVentana[3])+','+str(coorVentana[2])+
    ','+str(coorVentana[1])
    TIME = strftime( "%Y-%m-%d", gmtime())
    MAXCC = '10'

    url = 'https://services.sentinel-hub.com/ogc/wms/b7b5e3ef-5a40-4e2a-9fd3-75ca2b81cb32?SERVICE=WMS&REQUEST=GetMap&MAXCC='+MAXCC+'&LAYERS='+LAYERS+
    '&EVALSOURCE=S2&WIDTH=1000&HEIGHT=1000&ATMFILTER=ATMCOR&FORMAT=image/jpeg'+
    '&NICENAME=Sentinel-2+image+on+2019-01-30.jpg&TIME=2018-07-01/'+TIME+'&BBOX'+
    '='+BBOX+'&PREVIEW=3&EVALSCRIPT=cmV0dXJlIFtCMTIqMi41LEI4QSoyLjUsQjAzKjIuNV0%3D'

    r = requests.get(url)
    code = open(path+RGB+'/' + salida+str(x)+"_"+str(y)+".jpg", "wb")
    code.write(r.content)
    code = None
```

4. dataPuntosCalor.py contiene funciones para la generación y eliminación de los datos de entrada de puntos de calor de GOES-16.

```

import glob, os
import pandas as pd
import time as tm
from time import strptime, strftime

def copiaCSV():
    os.system('sh_/home/lanot/proceso/CSVPuntosCalor.sh')

def borraCSV():
    os.system('rm_/home/lanot/proceso/data.PuntosCalor/*')

def archivoCSV(path):
    csv = os.listdir(path)[0]
    return csv

def dataCSV(csv):
    df = pd.read_csv(csv)
    puntos = len(df.axes[0])
    x, y = [], []
    tiempoIni, tiempoTer = [], []
    csvArchivo = csv[csv.find('Calor')+6:]
    fechaPC = csvArchivo[csvArchivo.find('unam-')+5:csvArchivo.find('.csv')]
    fechaPC = strptime(fechaPC, '%m%d%H%M')
    fechaPC = strftime("%Y-%m-%d_%H:%M", fechaPC)

    for i in range(puntos):
        data = df.axes[0][i]
        if data[1] != 'longitude' and (-122.19 < float(data[1]) < -84.64) and
        (12.53 < float(data[0]) < 32.72):
            x.append(float(data[1])), y.append(float(data[0])), tiempoIni.append(data[2])
            tiempoTer.append(data[3])

    puntosMexico = len(x)
    coordenadas = {'lon':x, 'lat':y, 'tmpIni': tiempoIni, 'tmpFin': tiempoTer}
    return csv, puntos, coordenadas, csvArchivo, fechaPC, puntos, puntosMexico

def PuntosCalor(path):
    copiaCSV()
    csv = archivoCSV(path)
    csv, puntos, coordenadas, csvArchivo, fechaPC, puntos, puntosMexico = dataCSV(path+csv)
    borraCSV()
    return csv, puntos, coordenadas, csvArchivo, fechaPC, puntos, puntosMexico

```

### C.3. Procesamiento de datos

Scripts genéricos para el procesamiento de los datos específicos de cada satélite, con reducidas modificaciones en cada proceso.

1. GOES16.py contiene funciones el procesamiento de los datos de GOES-16.

```
import os
import genericSat as gs
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as crs
from osgeo import gdal, osr
from netCDF4 import Dataset

def archivosGOES16(path):
    archivos = []
    for i in range(7):
        if i+1 != 4:
            archivo = os.listdir(path+'C0'+str(i+1))[0]
            archivos.append(path+'C0'+str(i+1)+'/'+archivo)
    archivos.sort()
    return archivos

def verdeSinteticoG16(C01,C02,C03):
    verdeSint = 0.48358168 *C02 + 0.45706946 * C01 + 0.06038137 * C03
    return verdeSint

def regionGOES(data,region,nvl):
    if region == 'CONUS' and nvl == 'L1b':
        data = gs.rebin(data,[3000,5000])
    elif region == 'CONUS' and nvl == 'L2':
        data = gs.rebin(data,[1500,2500])
    return data

def extraeNetCDFL2(path,var,res):
    nc = Dataset(path)
    data = nc.variables[var][:].data
    banda = path[path.find('MBC')+3:path.find('_G16')]
    if banda == '01':
        data[data == 1023.] = np.nan
    elif banda == '02':
        data[data == 4095.] = np.nan
    elif banda == '03':
        data[data == 1023.] = np.nan
    elif banda == '05':
        data[data == 1023.] = np.nan
    elif banda == '06':
        data[data == 1023.] = np.nan
    elif banda == '07':
        data[data == 16383.0] = np.nan
    H = nc.variables['goes_imager_projection'].perspective_point_height
```

```

xmin = nc.variables['x_image_bounds'][0] * H
xmax = nc.variables['x_image_bounds'][1] * H
ymin = nc.variables['y_image_bounds'][1] * H
ymax = nc.variables['y_image_bounds'][0] * H
if res == 1:
    data = rebin(data , [3000,5000])
elif res == 2:
    data = rebin(data , [1500,2500])
nx,ny = data.shape[0],data.shape[1]
varName = nc.variables[ var ].long_name
stdName = nc.variables[ var ].standard_name
unidades = nc.variables[ var ].units
nc.close()
return data , xmin , ymin , xmax , ymax , nx , ny , varName , stdName , unidades

def FTGoes16(archivos , x , y , offset):
    coorVentana = gs.coordenadasVentana(x,y,offset)
    b7, xmin,ymin,xmax,ymax,nx,ny,varName,stdName,unidades
    = extraeNetCDFL2(archivos[5], 'CMI', 2)
    gs.creaTiff(b7, xmin, ymin, xmax, ymax, nx, ny)
    r = gs.dataVentana(x,y,offset)
    b6, xmin, ymin,xmax,ymax,nx,ny,varName,stdName,unidades
    = extraeNetCDFL2(archivos[4], 'CMI', 2)
    gs.creaTiff(b6, xmin, ymin, xmax, ymax, nx, ny)
    g = gs.dataVentana(x,y,offset)
    b5, xmin, ymin,xmax,ymax,nx,ny,varName,stdName,unidades
    = extraeNetCDFL2(archivos[3], 'CMI', 2)
    gs.creaTiff(b5, xmin,ymin, xmax, ymax, nx, ny)

    b = gs.dataVentana(x,y,offset)
    r,r = np.maximum(r, 273),np.minimum(r, 333)
    g,g = np.maximum(g, 0),np.minimum(g, 1)
    b,b = np.maximum(b, 0),np.minimum(b, .75)
    r,g,b = (r-273)/(333-273),(g-0)/(1-0),(b-0)/(.75-0)
    r = np.power(r, 2.5)
    rgb = gs.compuestoRGB(r,g,b,False)
    imagen = gs.plotRGB(rgb,coorVentana,x,y,'GOES16_FT.')
    return imagen

def TCGoes16(archivos , x , y , offset):
    coorVentana = gs.coordenadasVentana(x,y,offset)
    b3, xmin,ymin,xmax,ymax,nx,ny,varName,stdName,unidades
    = extraeNetCDFL2(archivos[1], 'CMI', 1)

    gs.creaTiff(b3, xmin, ymin, xmax, ymax, nx, ny)
    r = gs.dataVentana(x,y,offset)
    b2, xmin,ymin,xmax,ymax,nx,ny,varName,stdName,unidades
    = extraeNetCDFL2(archivos[2], 'CMI', 1)
    gs.creaTiff(b2, xmin, ymin, xmax, ymax, nx, ny)
    g = gs.dataVentana(x,y,offset)
    b1, xmin, ymin,xmax,ymax,nx,ny,varName,stdName,unidades
    = extraeNetCDFL2(archivos[0], 'CMI', 1)
    gs.creaTiff(b1, xmin, ymin, xmax, ymax, nx, ny)

```

```
b = gs.dataVentana(x,y,offset)
r,g,b = gs.correccionGamma(r,0.3),correccionGamma(g,0.1),correccionGamma(b,0.3)
r,g,b = gs.normaliza(r),normaliza(g),normaliza(b)
g = verdeSinteticoG16(b,r,g)
rgb = gs.compuestoRGB(r,g,b,True)
imagen = gs.plotRGB(rgb,coorVentana,x,y,'GOES16-TC_')
return imagen

def RGBGoes16(path,x,y,offset):
    archivos = archivosGOES16(path)
    TCimagen = TCGoes16(archivos,x,y,offset)
    FTimagen = FTGoes16(archivos,x,y,offset)
    return TCimagen,FTimagen
```

## 2. SuomiNPP.py contiene funciones el procesamiento de los datos de Suomi-NPP.

```
import os
import genericSat as gs
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as crs
from osgeo import gdal,osr

def archivosSuomiNPP(path):
    archivos = []

    for i in range(12):
        if 2 < i+1 < 6:
            tifs = os.listdir(path+'C0'+str(i+1))
            tifs.sort()
            tif_1,tif_2,tif_3 = tifs[0],tifs[1],tifs[2]
            archivos.append(path+'C0'+str(i+1)+'/'+tif_1)
            archivos.append(path+'C0'+str(i+1)+'/'+tif_2)
            archivos.append(path+'C0'+str(i+1)+'/'+tif_3)
        elif 9 < i+1 < 13:
            tifs = os.listdir(path+'C'+str(i+1))
            tifs.sort()
            tif_1,tif_2,tif_3 = tifs[0],tifs[1],tifs[2]
            archivos.append(path+'C'+str(i+1)+'/'+tif_1)
            archivos.append(path+'C'+str(i+1)+'/'+tif_2)
            archivos.append(path+'C'+str(i+1)+'/'+tif_3)
    return archivos

def verificaPaso(archivos):
    paso2,paso1 = archivos[1],archivos[2]
    diapasos2 = int(paso2[paso2.find('2019.')+7:paso2.find('2019.')+9])
    diapasos1 = int(paso1[paso1.find('2019.')+7:paso1.find('2019.')+9])
    if diapasos2 > diapasos1:
        paso = 2
    else:
        paso = 1
    return paso
```



```

def SuomiNPP_1paso(archivos):
    b3,b4,b5 = archivos[2],archivos[5],archivos[8]
    b10,b11,b12 = archivos[11],archivos[14],archivos[17]
    return b3,b4,b5,b10,b11,b12

def SuomiNPP_2pasos(archivos):
    b3_1,b3_2 = archivos[1],archivos[0]
    b4_1,b4_2 = archivos[4],archivos[3]
    b5_1,b5_2 = archivos[7],archivos[6]
    b10_1,b10_2 = archivos[10],archivos[9]
    b11_1,b11_2 = archivos[13],archivos[12]
    b12_1,b12_2 = archivos[16],archivos[15]
    return b3_1,b3_2,b4_1,b4_2,b5_1,b5_2,b10_1,b10_2,b11_1,b11_2,b12_1,b12_2

def paso2SuomiNPP(paso1,paso2):
    xmin,xmax,ymin,ymax = -118,-85,12,33.5
    ds_1,ds_2 = gdal.Open(paso1),gdal.Open(paso2)

    gdal.Warp('data_1_4326.tif',ds_1,options=gdal.WarpOptions(dstSRS='EPSG:4326',
    dstNodata=-9999))
    gdal.Warp('data_2_4326.tif',ds_2,options=gdal.WarpOptions(dstSRS='EPSG:4326',
    dstNodata=-9999))
    ds_1,ds_2=gdal.Open('data_1_4326.tif'),gdal.Open('data_2_4326.tif')
    gdal.Translate('data_1_4326_rec.tif',ds_1,options=gdal.TranslateOptions(projWin
    =[xmin,ymax,-94.5,ymin]))
    gdal.Translate('data_2_4326_rec.tif',ds_2,options=gdal.TranslateOptions(projWin
    =[-94.5,ymax,xmax,ymin]))

    ds_1,ds_2 = gdal.Open('data_1_4326_rec.tif'),gdal.Open('data_2_4326_rec.tif')
    data_1,data_2 = ds_1.ReadAsArray(),ds_2.ReadAsArray()
    data = np.concatenate((data_1,data_2),axis=1)
    data[data == -9999.000000] = np.nan
    data[data == -340282346638528859811704183484516925440.000000] = np.nan
    nx,ny = data.shape[0],data.shape[1]
    ds_1,ds_2,data_1,data_2 = None,None,None,None

    os.remove('data_1_4326.tif'),os.remove('data_2_4326.tif')
    os.remove('data_1_4326_rec.tif'),os.remove('data_2_4326_rec.tif')
    return data,xmin,ymin,xmax,ymax,nx,ny

def paso1SuomiNPP(paso1):
    xmin,xmax,ymin,ymax = -118,-85,12,33.5
    ds_1 = gdal.Open(paso1)

    gdal.Warp('data_1_4326.tif',ds_1,options=gdal.WarpOptions(dstSRS='EPSG:4326',
    dstNodata=-9999))
    ds_1 = gdal.Open('data_1_4326.tif')
    gdal.Translate('data_1_4326_rec.tif',ds_1,options=gdal.TranslateOptions(projWin
    =[xmin,ymax,xmax,ymin]))

    ds_1 = gdal.Open('data_1_4326_rec.tif')
    data = ds_1.ReadAsArray()

```

```
data[data == -9999.000000] = np.nan
data[data == -340282346638528859811704183484516925440.000000] = np.nan
nx,ny = data.shape[0],data.shape[1]
ds_1 = None

os.remove('data_1.4326.tif')
os.remove('data_1.4326.rec.tif')
return data, xmin, ymin, xmax, ymax, nx, ny

def FTSuomiNPP(archivos ,paso ,x,y,offset):
    coorVentana = gs.coordenadasVentana(x,y,offset)

    if paso == 2:
        archivos = SuomiNPP_2pasos(archivos)
        b12, xmin, ymin, xmax, ymax, nx, ny = paso2SuomiNPP(archivos[10],archivos[11])
        gs.creaTiff(b12, xmin, ymin, xmax, ymax, nx, ny)
        r = gs.dataVentana(coorVentana)
        b11, xmin, ymin, xmax, ymax, nx, ny = paso2SuomiNPP(archivos[8],archivos[9])
        gs.creaTiff(b11, xmin, ymin, xmax, ymax, nx, ny)
        g = gs.dataVentana(coorVentana)
        b10, xmin, ymin, xmax, ymax, nx, ny = paso2SuomiNPP(archivos[6],archivos[7])
        gs.creaTiff(b10, xmin, ymin, xmax, ymax, nx, ny)
        b = gs.dataVentana(coorVentana)
        r = np.power(r, 1.1)

    elif paso == 1:
        archivos = SuomiNPP_1paso(archivos)
        b12, xmin, ymin, xmax, ymax, nx, ny = paso1SuomiNPP(archivos[5])
        gs.creaTiff(b12, xmin, ymin, xmax, ymax, nx, ny)
        r = gs.dataVentana(coorVentana)
        b11, xmin, ymin, xmax, ymax, nx, ny = paso1SuomiNPP(archivos[4])
        gs.creaTiff(b11, xmin, ymin, xmax, ymax, nx, ny)
        g = gs.dataVentana(coorVentana)
        b10, xmin, ymin, xmax, ymax, nx, ny = paso1SuomiNPP(archivos[3])
        gs.creaTiff(b10, xmin, ymin, xmax, ymax, nx, ny)
        b = gs.dataVentana(coorVentana)
        r = np.power(r, 1.03)

    r,g,b = gs.normaliza(r),gs.normaliza(g),gs.normaliza(b)
    rgb = gs.compuestoRGB(r,g,b)
    imagen = gs.plotRGB(rgb,coorVentana,x,y,'SuomiNPP_FT_')
    return imagen

def TCSuomiNPP(archivos ,paso ,x,y,offset):
    coorVentana = gs.coordenadasVentana(x,y,offset)

    if paso == 2:
        archivos = SuomiNPP_2pasos(archivos)
        b5, xmin, ymin, xmax, ymax, nx, ny = paso2SuomiNPP(archivos[4],archivos[5])
        gs.creaTiff(b5, xmin, ymin, xmax, ymax, nx, ny)
        r = gs.dataVentana(coorVentana)
        b4, xmin, ymin, xmax, ymax, nx, ny = paso2SuomiNPP(archivos[2],archivos[3])
        gs.creaTiff(b4, xmin, ymin, xmax, ymax, nx, ny)
```

```

g = gs.dataVentana(coorVentana)
b3, xmin, ymin, xmax, ymax, nx, ny = paso2SuomiNPP(archivos[0], archivos[1])
gs.creaTiff(b3, xmin, ymin, xmax, ymax, nx, ny)
b = gs.dataVentana(coorVentana)
if x < -94.5 :
    gamma = 0.01
else :
    gamma = 0.005

elif paso == 1:
    archivos = SuomiNPP.lpaso(archivos)
    b5, xmin, ymin, xmax, ymax, nx, ny = paso1SuomiNPP(archivos[2])
    gs.creaTiff(b5, xmin, ymin, xmax, ymax, nx, ny)
    r = gs.dataVentana(coorVentana)
    b4, xmin, ymin, xmax, ymax, nx, ny = paso1SuomiNPP(archivos[1])
    gs.creaTiff(b4, xmin, ymin, xmax, ymax, nx, ny)
    g = gs.dataVentana(coorVentana)
    b3, xmin, ymin, xmax, ymax, nx, ny = paso1SuomiNPP(archivos[0])
    gs.creaTiff(b3, xmin, ymin, xmax, ymax, nx, ny)
    b = gs.dataVentana(coorVentana)
    gamma = 0.5

r, g, b = gs.correccionGamma(r, gamma), gs.correccionGamma(g, gamma), gs.
correccionGamma(b, gamma)
r, g, b = gs.normaliza(r), gs.normaliza(g), gs.normaliza(b)
rgb = gs.compuestoRGB(r, g, b)
imagen = gs.plotRGB(rgb, coorVentana, x, y, 'SuomiNPP_TC_')
return imagen

def RGBSuomiNPPI(path, x, y, offset):
    archivos = archivosSuomiNPP(path)
    paso = verificaPaso(archivos)
    TCimagen = TCSuomiNPP(archivos, paso, x, y, offset)
    FTimagen = FTSuomiNPP(archivos, paso, x, y, offset)
    return TCimagen, FTimagen, paso

```

### 3. Sentinel2.py contiene funciones el procesamiento de los datos de Sentinel2.

```
import os, dataSentinel2
import genericSat as gs
from osgeo import gdal
from pyproj import Proj, transform
import matplotlib.pyplot as plt
import cartopy.crs as crs

def archivoSentinel2(path):
    archivo = os.listdir(path)[0]
    return archivo

def reproyectaJPG(x,y, offset , archivo):
    coorVentana = dataSentinel2.coorVentanaSentinelHub(x,y, offset)
    ds = gdal.Open(archivo)
    gdal.Translate('tmp.tif', ds, options=gdal.TranslateOptions(outputBounds=
    coorVentana, outputSRS='EPSG:3857'))
    ds = gdal.Open('tmp.tif')
    gdal.Warp('tmp_4326.tif', ds, options=gdal.WarpOptions(dstSRS='EPSG:4326'))
    ds = None

def borraTmp():
    os.remove('tmp.tif')
    os.remove('tmp_4326.tif')

def RGBSentilen2(x,y, offset ,RGB, path , salida):
    coorVentana = gs.coordenadasVentana(x,y, offset)
    path = path+RGB
    archivo = archivoSentinel2(path)
    archivo = path+'/' +archivo
    reproyectaJPG(x,y, offset , archivo)
    imagen = gs.plotRGB(coorVentana ,x,y, salida)
    borraTmp()
    os.remove(archivo)
    return imagen
```

## C.4. Scripts secundarios

Scripts secundarios de los procesos desarrollados.

1. EMAS\_GOES16.py contiene funciones para la integración del producto TC de MSI/Sentinel-2 con LST de ABI/GOES-16.

```

import os, Sentinel2, dataSentinel2, dataGOES16, GOES16
import genericSat as gs
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as crs
from osgeo import gdal, osr
from netCDF4 import Dataset
from matplotlib.patches import Rectangle

def archivosGOES16(path):
    archivo = os.listdir(path)[0]
    archivo = path+archivo
    return archivo

def LSTCal(dataCal, x, y, offset, xmin, ymin, xmax, ymax, nx, ny):
    gs.creaTiff(dataCal, xmin, ymin, xmax, ymax, nx, ny)
    lstCal, lstCalValue = gs.dataVentana(x, y, offset)
    return lstCalValue

def LSTGoes16(archivos, x, y, offset):
    coorVentana = gs.coordenadasVentana(x, y, offset)
    data, dataCal, xmin, ymin, xmax, ymax, nx, ny, varName, stdName, unidades =
    GOES16.extraeNetCDFL2(archivos, 'LST', 2)
    gs.creaTiff(data, xmin, ymin, xmax, ymax, nx, ny)
    lst, lstValue = gs.dataVentana(x, y, offset)
    lstCalValue = LSTCal(dataCal, x, y, offset, xmin, ymin, xmax, ymax, nx, ny)
    imagen = gs.plotRGBbus(lst, lstValue, lstCalValue, coorVentana, x, y, offset,
    'GOES16_LST_', varName, stdName, unidades)
    return imagen, lstValue

def RGBGoes16(path, pathSen, x, y, offset):
    dataSentinel2.extraeWMS(x, y, offset, '1-NATURAL-COLOR,DATE', 'TC_', pathSen, 'TC')
    archivo = Sentinel2.RGBSentilen2(x, y, offset, 'TC', pathSen)
    archivos = archivosGOES16(path)
    LSTimagen, lstValue = LSTGoes16(archivos, x, y, offset)
    Sentinel2.borraTmp(), dataSentinel2.borraTodo(archivo)
    fechaG16, tiempoG16 = dataGOES16.revisaNetCDF()
    return LSTimagen, lstValue, fechaG16, tiempoG16

```

## 2. NOAA\_SuomiNPP.py contiene funciones para la integración del producto TC de MSI/Sentinel-2 con SST de VIIRS/Suomi-NPP

```
import os, Sentinel2, dataSentinel2, Sentinel2
import generciSat as gs
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as crs
from osgeo import gdal, osr
from matplotlib.patches import Rectangle

def archivosSuomiNPP(path):
    archivos = []
    tifs = os.listdir(path)
    tifs.sort()
    for i in tifs:
        archivos.append(path+i)
    return archivos

def SuomiNPP_1paso(archivos):
    sstTiff = archivos[0]
    return [sstTiff]

def SuomiNPP_2pasos(archivos):
    sstTiff_1, sstTiff_2 = archivos[2], archivos[1]
    return [sstTiff_1, sstTiff_2]

def SSTSuomiNPP(archivos, paso, x, y, offset):
    coorVentana = gs.coordenadasVentana(x, y, offset)
    if paso == 2:
        archivos = SuomiNPP_2pasos(archivos)
        sst, xmin, ymin, xmax, ymax, nx, ny = Sentinel2.paso2SuomiNPP(
            archivos[0], archivos[1])
        gs.creaTiff(sst, xmin, ymin, xmax, ymax, nx, ny)
        sst, sstValue = gs.dataVentana(coorVentana, x, y)
        sst[sst <= 20] = np.nan
    elif paso == 1:
        archivos = SuomiNPP_1paso(archivos)
        sst, xmin, ymin, xmax, ymax, nx, ny = Sentinel2.paso1SuomiNPP(archivos[0])
        gs.creaTiff(sst, xmin, ymin, xmax, ymax, nx, ny)
        sst, sstValue = gs.dataVentana(coorVentana, x, y)
    imagen = gs.plotRGB(sst, sstValue, coorVentana, x, y, offset, 'SuomiNPP_SST_', 'SST',
        'Sea_Surface_Temperature', 'C')
    return imagen, sstValue

def RGBSuomiNPP(path, pathSen, x, y, offset):
    dataSentinel2.extraeWMS(x, y, offset, '1-NATURAL-COLOR,DATE', 'TC_', pathSen, 'TC')
    archivo = Sentinel2.RGBSentilen2(x, y, offset, 'TC', pathSen)
    archivos = archivosSuomiNPP(path)
    paso = Sentinel2.verificaPaso(archivos)
    SSTimagen, SSTvalue = SSTSuomiNPP(archivos, paso, x, y, offset)
    Sentinel2.borraTmp()
    dataSentinel2.borraTodo(archivo)
    return SSTimagen, SSTvalue, paso
```

3. ubicacion.py contiene funciones para la generación del mapa de ubicación y extracción de información de la entidad y municipio.

```

import genericSat as gs
import geopandas as gpd
import matplotlib.pyplot as plt
import cartopy.crs as crs
from shapely.geometry import Point
from cartopy.mpl.gridliner import LONGITUDE_FORMATTER, LATITUDE_FORMATTER

def extSatelites(x,y):
    G16 = gs.coordenadasVentana(x,y,1)
    SNPP = gs.coordenadasVentana(x,y,0.5)
    Sen2 = gs.coordenadasVentana(x,y,0.1)
    return G16,SNPP,Sen2

def datosCoor(x,y,shpMuni,shpEnti):
    coord = Point(x,y)

    for i in range(len(shpMuni)):
        try:
            if shpMuni['geometry'][i].contains(coord) == True:
                muni = i
                nomMuni = shpMuni['NOMMUN'][i]
                nomEnt = shpMuni['NOM.ENT'][i]
                cveMun = shpMuni['CVE.MUN'][i]
                cveEnt = shpMuni['CVE.ENT'][i]
                cveGeo = shpMuni['CVEGEO'][i]
        except:
            print ("Incendio fuera de Rango")

    for i in range(len(shpEnti)):
        try:
            if shpEnti['geometry'][i].contains(coord) == True:
                enti = i
        except:
            print ("Incendio fuera de Mexico")

    return muni, enti, nomMuni, nomEnt, cveMun, cveEnt, cveGeo

def plotUbicaMuni(shpMuni, muni, nomMuni, x, y):
    f, ax = plt.subplots(1, figsize=(15, 10))

    ax.axis('off')
    ax = plt.axes(projection=crs.PlateCarree())
    ax = shpMuni.plot(axes=ax, color='#939393', linewidth=1, edgecolor='#2E2E2E')
    ax.set_extent([x-0.3,x+0.3,y-0.3,y+0.3])

    gl = ax.gridlines(draw_labels=True, linewidth=0.5,
                    color='black', alpha=0.8, linestyle='--')
    gl.xlabel_top = False
    gl.ylabel_right = False
    gl.xformatter = LONGITUDE_FORMATTER

```

```
gl.yformatter = LATITUDE_FORMATTER
gl.xlabel_style = {'size': 13, 'color': 'black'}
gl.ylabel_style = {'size': 13, 'color': 'black', 'rotation': 'vertical'}

try:
    if len(shpMuni['geometry'][muni]) > 1:
        for i in range(len(shpMuni['geometry'][muni])):
            plt.fill(shpMuni['geometry'][muni][i].exterior.xy[0],
                    shpMuni['geometry'][muni][i].exterior.xy[1], '#535353')

except:
    plt.fill(shpMuni['geometry'][muni].exterior.xy[0], shpMuni['geometry']
            [muni].exterior.xy[1], '#535353')

plt.plot(x, y, 'r.', markersize=20, label='Punto de calor')
ax.legend(loc='lower_left', fontsize=18, facecolor='white')
plt.text(shpMuni['geometry'][muni].centroid.x-0.05, shpMuni['geometry'][muni]
        .centroid.y, nomMuni.upper(), fontsize=18, fontname='gadugi', color='black')

plt.savefig('ubicacion_muni.png', transparent=True, dpi=300, bbox_inches=
            'tight', pad_inches=0)

def plotUbicaEdo(shpEnti, enti, nomEnt, G16, SNPP, Sen2):
    f, ax = plt.subplots(1, figsize=(15, 10), frameon=False)

    ax.axis('off')
    ax = plt.axes(projection=crrs.PlateCarree())
    ax.background_patch.set_facecolor('white')
    ax.set_extent([-119, -78, 11.5, 35.5])
    ax = shpEnti.plot(axes=ax, color='#939393', linewidth=0.5, edgecolor='white')
    ax.set_axis_off()

    try:
        if len(shpEnti['geometry'][enti]) > 1:
            for i in range(len(shpEnti['geometry'][enti])):
                plt.fill(shpEnti['geometry'][enti][i].exterior.xy[0],
                        shpEnti['geometry'][enti][i].exterior.xy[1], '#535353')

    except:
        plt.fill(shpEnti['geometry'][enti].exterior.xy[0], shpEnti['geometry']
                [enti].exterior.xy[1], '#535353')

    plt.fill([G16[0], G16[2], G16[2], G16[0], G16[0]], [G16[3], G16[3], G16[1], G16[1],
    G16[3]], 'b', alpha=0.5, label='GOES-16_/ABI')
    plt.fill([SNPP[0], SNPP[2], SNPP[2], SNPP[0], SNPP[0]], [SNPP[3], SNPP[3], SNPP[1],
    SNPP[1], SNPP[3]], 'r', alpha=0.5, label='Suomi-NPP_/VIIRS')
    plt.fill([Sen2[0], Sen2[2], Sen2[2], Sen2[0], Sen2[0]], [Sen2[3], Sen2[3], Sen2[1],
    Sen2[1], Sen2[3]], 'y', alpha=0.5, label='Sentinel-2_/MSI')
    ax.legend(loc='lower_left', facecolor='white', fontsize=18)

    plt.savefig('ubicacion_edo.png', dpi=300, bbox_inches='tight', pad_inches=0)
```



```
def ubicaPuntosCalor(x,y):  
    shpEnti = gpd.read_file('/home/lanot/procesos/dest2018gw/dest2018gw.shp')  
    shpMuni = gpd.read_file('/home/lanot/procesos/muni_2018gw/muni_2018gw.shp')  
    muni, enti, nomMuni, nomEnt, cveMun, cveEnt, cveGeo = datosCoor(x,y, shpMuni, shpEnti)  
    G16, SNPP, Sen2 = extSatelites(x,y)  
  
    plotUbicaMuni(shpMuni, muni, nomMuni, x, y)  
    plotUbicaEdo(shpEnti, enti, nomEnt, G16, SNPP, Sen2)  
return muni, enti, nomMuni, nomEnt, cveMun, cveEnt, cveGeo
```

4. ensamble.py contiene funciones para la generación del PNG de integración, con reducidas modificaciones en cada proceso.

```
import ubicacion
from PIL import Image, ImageDraw, ImageFont

def creaBase():
    base = Image.new('RGB', (10698, 5332), color = 'white')
    datosBase = Image.new('RGB', (3500, 4832), color = '#676767')
    return base, datosBase

def extImagenes(G16_TC_path, G16_FT_path, SNPP_TC_path, SNPP_FT_path,
Sen2_TC_path, Sen2_SWIR_path, ubiMun_path, ubiEnt_path, logo_path):

    G16_TC = Image.open(G16_TC_path)
    G16_FT = Image.open(G16_FT_path)
    SNPP_TC = Image.open(SNPP_TC_path)
    SNPP_FT = Image.open(SNPP_FT_path)
    Sen2_TC = Image.open(Sen2_TC_path)
    Sen2_SWIR = Image.open(Sen2_SWIR_path)
    ubiMun = Image.open(ubiMun_path)
    ubiEdo = Image.open(ubiEnt_path)
    logo = Image.open(logo_path)

    return G16_TC, G16_FT, SNPP_TC, SNPP_FT, Sen2_TC, Sen2_SWIR,
ubiMun, ubiEdo, logo

def pegaImagenes(base, datosBase, G16_TC, G16_FT, SNPP_TC, SNPP_FT, Sen2_TC,
Sen2_SWIR, ubiMun, ubiEnt, logo)

    G16_TC = G16_TC.crop((10, 10, G16_TC.width - 10, G16_TC.height - 10))
    G16_FT = G16_FT.crop((10, 10, G16_FT.width - 10, G16_FT.height - 10))
    SNPP_TC = SNPP_TC.crop((10, 10, SNPP_TC.width - 10, SNPP_TC.height - 10))
    SNPP_FT = SNPP_FT.crop((10, 10, SNPP_FT.width - 10, SNPP_FT.height - 10))
    Sen2_TC = Sen2_TC.crop((10, 10, Sen2_TC.width - 10, Sen2_TC.height - 10))
    Sen2_SWIR = Sen2_SWIR.crop((10, 10, Sen2_SWIR.width - 10,
Sen2_SWIR.height - 10))

    base.paste(G16_TC, (50, 300))
    base.paste(G16_FT, (50, 2816))
    base.paste(SNPP_TC, (2416, 300))
    base.paste(SNPP_FT, (2416, 2816))
    base.paste(Sen2_TC, (4782, 300))
    base.paste(Sen2_SWIR, (4782, 2816))

    ubiMun = ubiMun.resize((1280, 1250), Image.ANTIALIAS)
    ubiEnt = ubiEnt.resize((3494, 2316), Image.ANTIALIAS)

    ubiEnt = ubiEnt.crop((10, 10, ubiEnt.width - 10, ubiEnt.height - 10))
    ubiMun = ubiMun.crop((1, 1, ubiMun.width - 1, ubiMun.height - 1))

    base.paste(ubiEnt, (7148, 300))
    base.paste(ubiMun, (9375, 300), ubiMun)
```

```

def regionCoordenadas(x,y):
    if x < 0 and y > 0 :
        xNom,yNom = 'W', 'N'
    elif x > 0 and y > 0 :
        xNom,yNom = 'E', 'N'
    elif x < 0 and y < 0 :
        xNom,yNom = 'W', 'S'
    else :
        xNom,yNom = 'E', 'S'
    coorFor = str("%0.2f" % abs(round(x,2)))+ '_' +xNom+ '_' +str("%0.2f" %
    abs(round(y,2)))+ '_' +yNom
    return coorFor

def aditexto(base ,muni ,enti ,nomMuni ,nomEnt ,cveMun ,cveEnt ,
cveGeo , coorFor ,tmpPC , fechaG16 , fechaSNPP):

    font = ImageFont.truetype('gadugi-2.ttf', size=150)
    satellite = ImageDraw.Draw(base)
    satellite.text((500,50), "GOES-16/Sensor_ABI", font=font , fill='black')
    satellite.text((2750,50), "Suomi-NPP/Sensor_VIIRS", font=font , fill='black')
    satellite.text((5200,50), "Sentinel-2/Sensor_MSI", font=font , fill='black')
    satellite.text((8450,50), "UBICACION", font=font , fill='black')

    font = ImageFont.truetype('gadugi-2.ttf', size=80)
    compuesto = ImageDraw.Draw(base)
    compuesto.text((185,2650), "True_Color_[C01(0.47_µm), C02(0.64_µm), C03(0.86_µm)]"
    , font=font , fill='black')
    compuesto.text((2600,2650), "True_Color_[C03(0.48_µm), C04(0.55_µm), C05(0.67_µm)]"
    , font=font , fill='black')
    compuesto.text((5000,2650), "True_Color_[C02(0.49_µm), C03(0.55_µm), C04(0.66_µm)]"
    , font=font , fill='black')
    compuesto.text((185,5160), "Fire_Temperature_[C05(1.6_µm), C06(2.25_µm), C07(3.9_µm)]"
    , font=font , fill='black')
    compuesto.text((2550,5160), "Fire_Temperature_[C10(1.61_µm), C11(2.25_µm), C12(3.7_µm)]"
    , font=font , fill='black')
    compuesto.text((5100,5160), "SWIR_[C03(0.55_µm), C8A(0.83_µm), C12(2.2_µm)]"
    , font=font , fill='black')

    font = ImageFont.truetype('gadugi-2.ttf', size=125)
    puntoCalor = ImageDraw.Draw(base)
    puntoCalor.text((7950,2700), 'Punto_de_Calor_GOES-16_detectado',
    font=font , fill='black')
    puntoCalor.text((8250,2725), '\n'+tmpPC+'_GMT', font=font , fill='black')

    font = ImageFont.truetype('gadugi-2.ttf', size=250)
    coordenada = ImageDraw.Draw(base)
    coordenada.text((8000,3000), coorFor , font=font , fill='black')

    font = ImageFont.truetype('gadugi-2.ttf', size=120)
    ubica = ImageDraw.Draw(base)
    ubica.text((7180,3400), 'LOCALIZACION\nMunicipio:\nEntidad:\nClave_Municipio '+
    ':\nClave_Entidad:\nClave_Geostatistica:', font=font , fill='black')
    ubica.text((9000,3400), '\n'+nomMuni+' \n'+nomEnt+' \n'+cveMun+' \n'+cveEnt+' \n'

```

```
+cveGeo, font=font, fill='black')

font = ImageFont.truetype('gadugi-2.ttf', size=120)
tiempos = ImageDraw.Draw(base)
tiempos.text((7180,4400), 'TIEMPO_DE_ESCANEO\nGOES-16/ABI:\nSuomi-NPP/VIIRS '+
'\nSentinel-2/MSI', font=font, fill='black')
tiempos.text((9000,4400), '\n'+fechaG16+' _GMT\n'+fechaSNPP+' _GMT\n-',
font=font, fill='black')

def ensambleSat(x, y, G16_TC_path, G16_FT_path, SNPP_TC_path, SNPP_FT_path, Sen2_TC_path
, Sen2_SWIR_path, ubiMun_path, ubiEnt_path, logo_path, tmpPC, fechaG16, fechaSNPP):

    muni, enti, nomMuni, nomEnt, cveMun, cveEnt, cveGeo = ubicacion.ubicaPuntosCalor(x, y)

    base, datosBase = creaBase()

    G16_TC, G16_FT, SNPP_TC, SNPP_FT, Sen2_TC, Sen2_SWIR, ubiMun, ubiEdo, logo = extImagenes
(G16_TC_path, G16_FT_path, SNPP_TC_path, SNPP_FT_path, Sen2_TC_path, Sen2_SWIR_path
, ubiMun_path, ubiEnt_path, logo_path)

    pegaImagenes(base, datosBase, G16_TC, G16_FT, SNPP_TC, SNPP_FT, Sen2_TC, Sen2_SWIR,
ubiMun, ubiEdo, logo)
    coordFor = regionCoordenadas(x, y)
    aditexto(base, muni, enti, nomMuni, nomEnt, cveMun, cveEnt, cveGeo, coordFor, tmpPC,
fechaG16, fechaSNPP)

    base.save('/var/www/html/incendios/Incendio'+str(x)+'_'+str(y)+'.jpg')
    base = None
    datosBase = None
```

## C.5. Scripts principales

Scripts principales de los procesos desarrollados.

### 1. EMAS.py contiene la función principal del proceso

```

import os , urllib3 ,EMAS_GOES16 ,dataGOES16 ,ensambleCoor
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

def url_xls(url):
    http = urllib3.PoolManager()
    r = http.request('GET', url, preload_content=False)
    filename = url[url.rfind("/")+5:-4]
    f = open(filename+'.xls', "wb")
    f.write(r.read())
    f.close(), r.close()
    return filename

def xls_dataframe(filename):
    data_xls = pd.read_excel(filename + ".xls", index_col=None)
    data_xls.to_csv(filename+'.csv', encoding='utf-8', index=False)
    df=pd.read_csv(filename+'.csv', sep=',', header=None)
    os.remove(filename + ".xls")
    df=pd.read_csv(filename+'.csv', sep=',', header=None)
    os.remove(filename+'.csv')
    return df

def datosEstacion(df):
    estacion ,ubicacion ,operador ,altitud = df.iat[0,1],df.iat[0,2],df.iat[1,1],df.iat[2,5]
    lon = df.iloc[2,1]
    lonDeg = float(lon[:lon.find('\xc2\x00')])
    lonMin = float(lon[lon.find('\xc2\x00')+2:lon.find('\xc2\x00')+4])
    lonSeg = float(lon[lon.find('\xc2\x00')+5:lon.find('\xc2\x00')+7])
    lon = lonDeg + lonMin/60 + lonSeg/3600
    lat = df.iloc[2,3]
    latDeg = float(lat[:lat.find('\xc2\x00')])
    latMin = float(lat[lat.find('\xc2\x00')+2:lat.find('\xc2\x00')+4])
    latSeg = float(lat[lat.find('\xc2\x00')+5:lat.find('\xc2\x00')+7])
    lat = latDeg + latMin/60 + latSeg/3600
    return estacion ,ubicacion ,operador ,altitud ,-lon , lat

def extraeVariables(df):
    temperatura = df.iloc[5:,5].values
    temperatura = temperatura.astype(float)
    return temperatura

```

```
def extraeTiempo(df):
    df[['fecha', 'hora']] = df[0].str.split('_', n=1, expand=True)
    df[['horas', 'minutos']] = df['hora'].str.split(':', n=1, expand=True)
    df[['min', 'segundos']] = df['minutos'].str.split(':', n=1, expand=True)
    fecha = df.iloc[5:,10].values
    hora = df.iloc[5:,11].values
    horas = df.iloc[5:,12].values
    minutos = df.iloc[5:,14].values
    return fecha, hora, horas, minutos

def descargaEMAS(estacion):
    url = "http://smn1.conagua.gob.mx/emas/exc/"+estacion+"_10M.xls"
    filename = url_xls(url)
    df = xls_dataframe(filename)
    df = df.replace(np.nan, 0)
    estacion, ubicacion, operador, altitud, lon, lat = datosEstacion(df)
    temperatura = extraeVariables(df)
    fecha, hora, horas, minutos = extraeTiempo(df)
    return estacion, ubicacion, operador, altitud, lon, lat, temperatura, fecha, hora,
    horas, minutos

def EMAS_LST(estaciones, pathLST, pathSen, pathEMAS, pathEMAS_LST, pathSalida, offset):
    dataGOES16.extraeNetCDF()

    for i in estaciones:
        try:
            estacion, ubicacion, operador, altitud, lon, lat, temperatura, fecha,
            hora, horas, minutos = descargaEMAS(i)
            LSTimagen, lstValue, fechaG16, tiempoG16 = EMAS_GOES16.RGBGoes16(pathLST,
            pathSen, lon, lat, offset)
            tiempoUG16 = tiempoG16
            lstUValue = lstValue
            data_lst_EMAS = open(pathEMAS+i+'.csv', 'a')
            data_lst_EMAS.write('\n'+str(lstValue)+' '+str(tiempoG16))
            data_lst_EMAS.close()
            csv_lst_emas = pd.read_csv(pathEMAS+i+'.csv')
            lstValue24 = csv_lst_emas.iloc[0:,0].values.tolist()

            if len(lstValue24) >= 24:
                csv_lst_emas = csv_lst_emas.iloc[1:]
                csv_lst_emas.to_csv(pathEMAS+i+'.csv', index=False)
                csv_lst_emas = pd.read_csv(pathEMAS+i+'.csv')
                lstValue = csv_lst_emas.iloc[0:,0].values
                tiempoG16 = csv_lst_emas.iloc[0:,1].values
                csv_lst_emas = None
                tiempo = []

            for j in horas+minutos:
                tiempo.append(float(j))
            cont = 0
            for j in tiempo:
                if float(tiempoUG16)-10 <= j <= float(tiempoUG16)+10:
                    break
                cont = cont + 1
```

```

data_lst_EMAS = open(pathEMAS_LST+i+'.csv','a')
data_lst_EMAS.write('\n'+i+', '+str(lstUValue)+' ,'+
+str(temperatura[cont])+' ,'+str(tiempoUG16)+' ,'+
+str(tiempo[cont])+' ,'+str(lon)+' ,'+str(lat))
data_lst_EMAS.close()
data_lst_EMAS EMC = open(pathEMAS_LST+'EMAS.lst.csv','a')
data_lst_EMAS EMC.write('\n'+i+', '+str(lstUValue)+' ,'+str(temperatura
[cont])+' ,'+str(tiempoUG16)+' ,'+str(tiempo[cont])+' ,'+str(lon)+' ,'+
+str(lat)+' ,'+str(altitud))
data_lst_EMAS EMC.close()
gs.creaGrafica(200,2,tiempo,tiempoG16,temperatura,lstValue,'Temperatura',
'(C)', 'r.',estacion,ubicacion)
ensembleCoord.ensembleSat(lon,lat,LSTimagen,'grafica.png',
'ubicacion.muni.png','ubicacion.edo.png','LANOT.png',pathSalida,estacion,
ubicacion,operador,altitud)
except:
    print 'Estacion sin transmision de datos'
dataGOES16.borraNetCDF()

def main():
estaciones = ('QR04','YC03','CM04','TB07','CS05','OX03','VR07','PU03','GR08',
'MO01','TL01','DF05','HI03','MX04','MC02','QO01','GT01','JA05','AG01','ZC01',
'NY03','SL02','SI04','DG05','TM04','NL01','CL16','CH22','SO08','BC06','BS16')

pathLST = '/home/lanot/proceso/data_GOES16/LST/'
pathSen = '/home/lanot/proceso/data_Sentine12/'
pathEMAS = '/home/lanot/proceso/data_EMAS_LST/'
pathEMAS_LST = '/home/lanot/proceso/data_EMAS_LST EMC/'
pathSalida = '/var/www/html/emas_lst/'
offset = 0.3
EMAS_LST(estaciones,pathLST,pathSen,pathEMAS,pathEMAS_LST,pathSalida,offset)

if __name__ == '__main__':
    main()

```

## 2. Boyas\_NOAA.py contiene la función principal del proceso 2.

```
import os, urllib3, requests, ensembleCoor, NOAA_SuomiNPP, dataSuomiNPP
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from bs4 import BeautifulSoup

def metaNOAA(url):
    r = requests.get("https://" + url)
    data = r.text
    status_code = r.status_code
    if status_code != 200:
        print ('Error_de_descarga...')
        return
    soup = BeautifulSoup(data)
    metadataText = []
    metadataDatos = []
    for metadata in soup.find_all('b'):
        metadataText.append(metadata.text)
    for metadata in soup.find_all('h1'):
        boyaDatos = metadata.text
    for metadata in soup.find_all('td'):
        metadataDatos.append(metadata.text)
    boyaNum = str(boyaDatos[boyaDatos.find("Station")+8:
    boyaDatos.find("Station")+13])
    ubicacion = str(boyaDatos[boyaDatos.find("-")+2:])
    lat = float(metadataText[3][:6])
    lon = float("-"+metadataText[3][9:15])
    prof = str(metadataDatos[15][288:291])
    return boyaNum, ubicacion, lat, lon, prof

def datosBoya(url):
    http = urllib3.PoolManager()
    r = http.request('GET', url, preload_content=False)
    filename = url[url.rfind("y2")+3:]
    f = open(filename, "wb")
    f.write(r.read())
    f.close(), r.close()
    df = pd.read_csv(filename, header=None, delimiter=r"\s+")
    os.remove(filename)
    WIMP = df.iloc[2:80,14].values
    ano = df.iloc[2:80,0].values
    mes = df.iloc[2:80,1].values
    dia = df.iloc[2:80,2].values
    hora = df.iloc[2:80,3].values
    minuto = df.iloc[2:80,4].values
    fecha = ano+"-"+mes+"-"+dia
    tiempo = hora+minuto
    tiempo = tiempo.astype(float)
    return WIMP, fecha, tiempo
```



```

def depurador(WIMP,time):
    WIMP[WIMP=='MM'] = np.nan
    WIMP = WIMP.astype(float)
    WIMP = np.flipr([WIMP])[0]
    time = np.flipr([time])[0]
    return WIMP,time

def descargaBoyaNOAA(boya):
    url1 = "www.ndbc.noaa.gov/station_page.php?station="+boya
    boyaNum,ubicacion,lat,lon,prof = metaNOAA(url1)
    url2 = "https://www.ndbc.noaa.gov/data/5day2/"+boya+"_5day.txt"
    WIMP,fecha,tiempo = datosBoya(url2)
    WIMP,tiempo = depurador(WIMP,tiempo)
    return boyaNum,ubicacion,lat,lon,prof,WIMP,fecha,tiempo

def Boya_SST(estaciones,pathSST,pathSen,pathEMAS,pathSalida,offset):
    dataSuomiNPP.extraeGTiff()
    for i in estaciones:
        boyaNum,ubicacion,lat,lon,prof,WIMP,fecha,tiempo = descargaBoyaNOAA(i)
        SSTimagen,SSTvalue,paso = NOAA.SuomiNPP.RGBSuomiNPP(pathSST,pathSen,
            lon,lat,offset)
        fechaSNPP,tiempoSNPP = dataSuomiNPP.reviseGTif(paso)
        creaGrafica(200,0.1,tiempo,tiempoSNPP,WIMP,SSTvalue,
            'Temperatura_Superficial_del_Mar','C','r.')
        ensambleCoor.ensambleSat(lon,lat,SSTimagen,'grafica.png','ubicacion_muni.png',
            'ubicacion_edo.png','LANOT.png',pathSalida,boyaNum,ubicacion,prof)
    dataSuomiNPP.borraGTiff()

def main():
    boyas = ['42055','42020','42019','42035','42003']
    pathSST = '/home/lanot/proceso/data_SuomiNPP/SST/'
    pathSen = '/home/lanot/proceso/data_Sentine12/'
    pathEMAS = '/home/lanot/proceso/data_BNOAA/'
    pathSalida = '/var/www/html/boyas_sst/'
    offset = 0.3
    Boya_SST(boyas,pathSST,pathSen,pathEMAS,pathSalida,offset)

if __name__ == '__main__':
    main()

```

### 3. PuntosCalor.py contiene la función principal del proceso 3.

```
import os, dataPuntosCalor, dataGOES16, dataSuomiNPP, dataSentinel2,
GOES16, SuomiNPP, Sentinel2, ensamble

def main():
    path = '/home/lanot/proceso/'
    pathPuntosCalor = path+'data_PuntosCalor/'
    pathGOES16 = path+'data_GOES16/'
    pathSuomiNPP = path+'data_SuomiNPP/'
    pathSentinel2 = path+'data_Sentinel2/'
    csv, puntos, coordenadas, csvArchivo, fechaPC, puntos, puntosMexico =
    dataPuntosCalor.PuntosCalor(pathPuntosCalor)

    info = 'Archivo_CSV:_'+csvArchivo+'\n'+ 'Fecha_CSV:_'+fechaPC+'\n'
    +'Puntos_de_Calor_detectados:_', str(puntos), '\n'+
    'Puntos_de_Calor_detectados_Mexico:_', str(puntosMexico),
    '\n'+'\n'+'\n'+'\n'+ 'lon, lat, tmpFinPC '
    infoArchivo = open('/var/www/html/incendios/info_'+csvArchivo[:-3]+
    'txt', 'a')
    infoArchivo.writelines(info)

    if len(coordenadas['lon']) == 0:
        infoArchivo.write('\nNo_se_detectaron_incendios_')
        infoArchivo.close()
    else:
        infoArchivo.close()
        dataPuntosCalor.plotPuntos(coordenadas['lon'], coordenadas['lat'])
        dataGOES16.extraeNetCDF()
        dataSuomiNPP.extraeGTiff()
        for x,y,tmpPC in zip(coordenadas['lon'], coordenadas['lat'],
        coordenadas['tmpFin']):

            try:
                infoArchivo = open('/var/www/html/incendios/info_'+csvArchivo
               [:-3]+'txt', 'a')
                offsetGOES16 = 1
                offsetSuomiNPP = 0.5
                offsetSentinel2 = 0.1

                dataSentinel2.extraeWMS(x,y, offsetSentinel2, '1-NATURAL-COLOR,DATE'
                , 'TC_', pathSentinel2, 'TC')
                dataSentinel2.extraeWMS(x,y, offsetSentinel2,
                '2-COLOR-INFRARED-VEGETATION-,DATE', 'SWIR_', pathSentinel2, 'SWIR')

                G16_TC_imagen, G16_FT_imagen = GOES16.RGBGoes16(pathGOES16, x, y,
                offsetGOES16)
                SNPP_TC_imagen, SNPP_FT_imagen, paso = SuomiNPP.RGBSuomiNPP(
                pathSuomiNPP, x, y, offsetSuomiNPP)
                Sen2_TC_imagen = Sentinel2.RGBSentilen2(x,y, offsetSentinel2, 'TC', p
                athSentinel2, 'Sentinel2_TC_')
                Sen2_SWIR_imagen = Sentinel2.RGBSentilen2(x,y, offsetSentinel2, 'SWIR',
                pathSentinel2, 'Sentinel2_SWIR_')

                fechaG16 = dataGOES16.revisaNetCDF('1')
```

```
if paso == 1:
    fechaSNPP = dataSuomiNPP.revisaGTif('03', '19')
else:
    fechaSNPP = dataSuomiNPP.revisaGTif('03', '18')

logo_path = 'logo.jpg'
ubiMun_path = 'ubicacion_muni.png'
ubiEnt_path = 'ubicacion_edo.png'
ensamble.ensambleSat(x,y,G16_TC.imagen,G16_FT.imagen,
SNPP_TC.imagen,SNPP_FT.imagen
,Sen2_TC.imagen,Sen2_SWIR.imagen,ubiMun_path,ubiEnt_path,
logo_path,tmpPC,fechaG16,fechaSNPP)

infoArchivo.write('\n'+str(x)+' '+str(y)+' '+str(tmpPC))
infoArchivo.close()

os.remove(G16_TC.imagen)
os.remove(G16_FT.imagen)
os.remove(SNPP_TC.imagen)
os.remove(SNPP_FT.imagen)
os.remove(Sen2_TC.imagen)
os.remove(Sen2_SWIR.imagen)
os.remove('ubicacion_edo.png')
os.remove('ubicacion_muni.png')
os.remove('tmp_rec.tif.aux.xml')
os.remove('tmp_4326_rec.tif.aux.xml')

except:
    print ("Incendio_fuera_de_rango")

dataGOES16.borraTodo()
dataSuomiNPP.borraTodo()
dataSentinel2.borraTodo()

if __name__ == '__main__':
    main()
```

## Apéndice D

# Sentencias Cron

Sentencias cron utilizadas para la automatización de los procesos principales.

1. Sentencia Cron que ejecuta el script al minuto 0 cada hora.

```
0 * * * * cd /home/lanot/proceso1;python EMAS.py
```

2. Sentencia Cron que ejecuta el script al minuto 0 a las 12:00.

```
0 12 * * * cd /home/lanot/proceso2;python Boyas.NOAA.py
```

3. Sentencia Cron que ejecuta el script al minuto 0 de las 10:00 a las 18:00.

```
0 10-18 * * * cd /home/lanot/proceso3;python PuntosCalor.py
```



# Bibliografía

- [Alvarez,2012] ALVAREZ J. ¿Qué es GMT (Generic Mapping Tools)?. (2012, Junio 20). Recuperado el 13 de Diciembre del 2018 de <https://josealvarezgomez.wordpress.com/2011/05/28/que-es-gmt-generic-mapping-tools/>
- [Bense,2007] BENSE T. (2007). *Introducción a la percepción remota. Combinaciones de color*. Sextas Jornadas de Educación en Percepción Remota en el Ambito del Mercosur y Primeras Uruguayas. Recuperado el 1 de Enero del 2019 de <http://www.teledet.com.uy/tutorial-imagenes-satelitales/combinaciones-colores.html>
- [Chuvieco,1995] CHUVIECO, E. (1995). *Fundamentos de teledeteccion espacial*. (2da ed). Madrid: Rialp. 89-114
- [EARTHDATA,2019] Earthdata/GeoTIFF. (2019). Recuperado el 14 de Enero del 2019 de <https://earthdata.nasa.gov/user-resources/standards-and-references/geotiff>
- [Eduspace,2011] Eduspace ES - Inicio - Los satélites de recursos naturales.(2011, Enero 26). Recuperado el 20 de Diciembre del 2018 de [http://www.esa.int/SPECIALS/Eduspace\\_ES/SEM4Y4E3GXF\\_0.html](http://www.esa.int/SPECIALS/Eduspace_ES/SEM4Y4E3GXF_0.html)
- [Flowerdew,1991] FLOWERDEW, R. (1991) *Spatial data integration. Geographical information systems*. Vol. 1. 375-387.
- [GDAL,2018] GDAL/OGR contributors. User Oriented Documentation. (2018). Recuperado el 8 de Noviembre del 2018 de <http://gdal.org/>
- [Godin y Gottshall,2013] GODIN R. y GOTTSHALL E. (2013). *Joint Polar Satellite System (JPSS) VIIRS Sea Surface Temperature Algorithm Theoretical Basis Document (ATBD)*. Joint Polar Satellite System (JPSS) Ground Project. 1-16. Recuperado el 29 de Diciembre del 2018 de [https://www.star.nesdis.noaa.gov/jpss/documents/ATBD/D0001-M01-S01-010\\_JPSS\\_ATBD\\_VIIRS-SST\\_A.pdf](https://www.star.nesdis.noaa.gov/jpss/documents/ATBD/D0001-M01-S01-010_JPSS_ATBD_VIIRS-SST_A.pdf)
- [Godin y Vicente,2013] GODIN R. y VICENTE G. (2013). *Joint Polar Satellite System (JPSS) Operational Algorithm Description (OAD) Document for VIIRS Land Surface Temperature (LST) Environmental Data Records (EDR) Software*. Joint Polar Satellite System (JPSS) Ground Project. 4-15. Recuperado el 28 de

Diciembre del 2018 de [https://jointmission.gsfc.nasa.gov/sciencedocs/2017-06/474-00070\\_0AD-VIIRS-LST-EDR\\_E.pdf](https://jointmission.gsfc.nasa.gov/sciencedocs/2017-06/474-00070_0AD-VIIRS-LST-EDR_E.pdf)

[GOES-R-eoPortal,n.f] GOES-R - eoPortal Directory - Satellite Missions. (n.f). Recuperado el 21 de Diciembre del 2018 de <https://directory.eoportal.org/web/eoportal/satellite-missions/g/goes-r>

[GOES-16,2018] GOES-16. (2018, Septiembre 04). Recuperado el 23 de Diciembre del 2018 de <https://en.wikipedia.org/wiki/GOES-16>

[González,2010] GONZÁLEZ-MARENTES H. (2010). Los satélites meteorológicos y ambientales como herramientas de trabajo operativas para la meteorología, la hidrología y la oceanología en Colombia y en apoyo a las actividades de prevención de desastres. *En Experiencias en el uso y aplicación de tecnologías satelitales para observación de la Tierra*. Bogotá D. C., Colombia : El Instituto. 46-47

[Ignatov,2010] IGNATOV A. (2010). *GOES-R Advanced Baseline Imager (ABI) Algorithm Theoretical Basis Document for Sea Surface Temperature*. NOAA NESDIS center for satellite applications and research. 14-19, 30-45. Recuperado el 1 de Enero del 2019 de <https://www.goes-r.gov/products/ATBDs/baseline/baseline-SST-v2.0.pdf>

[Lawhead,2015] LAWHEAD, J. (2015) *Learning geospatial analysis with Python: An effective guide to geographic information system and remote sensing analysis using Python 3*.(2da ed). Birmingham: Packt Publishing.

[Lindstrom,Bah,Schmit y Kohrs,n.f ] LINDSTROM S.,BAH K., SCHMIT T. y KOHRS R. (n.f). *CIMSS Natural True Color, Quick Guide*. Recuperado el 5 de Enero del 2019 de [https://www.star.nesdis.noaa.gov/GOES/documents/ABIQuickGuide\\_CIMSSRGB\\_v2.pdf](https://www.star.nesdis.noaa.gov/GOES/documents/ABIQuickGuide_CIMSSRGB_v2.pdf)

[LOC,2012] LOC. Sustainability of Digital Formats: Planning for Library of Congress Collections. (2012). Recuperado el 12 de Enero del 2019 de <https://www.loc.gov/preservation/digital/formats/fdd/fdd000332.shtml>

[Luque y Amengual,n.f] LUQUE A. y AMENGUAL B. *Teledetección: Aplicaciones Meteorológicas*. España : Universitat de les Illes Balears, Departament de Física, Grup de Meteorología. 3-9

[Miller,2003] MILLER E., GARANT M., ORINGER L., REINING R., SHAFFER D., STERLING J. y TAUB A.(10-14 de noviembre de 2003). *Satélite geoestacionario de próxima generación (GOES- R)*. Beneficios económicos de la Administración Nacional Oceánica y Atmosférica (NOAA). NOAA. Honolulu, Hawai

[NOAA,2018] NOAA. About Our Agency. (2018). Recuperado el 5 de Noviembre del 2018 de <https://www.noaa.gov/about-our-agency>

[Norman y Becker,n.f] NORMAN J. y BECKER F. *Terminology in thermal infrared remote sensing of natural surfaces*. Agricultural and Forest Meteorology. Vol. 77. 153-154. Recuperado el 25 de Diciembre del 2018 de <https://www.sciencedirect.com/science/article/pii/016819239502259Z>.

- [Órbita heliosíncrona,2018] Órbita heliosíncrona.(2018, Noviembre 18). Recuperado el 20 de Diciembre del 2018 de [https://es.wikipedia.org/wiki/Orbita\\_heliosincrona](https://es.wikipedia.org/wiki/Orbita_heliosincrona)
- [PODACC,n.f] PODACC. GOES16-OSISAF-L3C-v1.0. (n.f). Recuperado el 30 de Diciembre del 2018 de <https://podaac.jpl.nasa.gov/dataset/GOES16-OSISAF-L3C-v1.0?ids=Sensor:Platform&values=ABI:GOES-16>
- [Python,2018] Python. (2018, Diciembre 06). Recuperado el 15 de Diciembre del 2018 de <https://es.wikipedia.org/wiki/Python>
- [RAMMB,n.f] RAMMB. NASA SPoRT. (n.f). Fire Temperature RGB, Quick Guide. Recuperado el 6 de Enero del 2019 de [http://rammb.cira.colostate.edu/training/visit/quick\\_guides/Fire\\_Temperature\\_RGB.pdf](http://rammb.cira.colostate.edu/training/visit/quick_guides/Fire_Temperature_RGB.pdf)
- [Rodríguez,Benito y Portela,2004] RODRÍGUEZ-JIMÉNEZ R.M., BENITO-CAPA A. y PORTELA-LOZANO A. (2004). Los mapas meteorológicos. *En Unidad didáctica. Meteorología y Climatología*. España: Fundación española para la ciencia y la tecnología. 50-51
- [Royle,2011] ROYLE A. W .(5-12 de marzo de 2011). *Visión general de la arquitectura del segmento terrestre del GOES-R*. Conferencia Aeroespacial IEEE 2011. Big Sky, MT, EUA.
- [Satélite meteorológico,2018] Satélite meteorológico. (2018, Diciembre 03). Recuperado el 18 de Diciembre del 2018 de [https://es.wikipedia.org/wiki/Satelite\\_meteorologico](https://es.wikipedia.org/wiki/Satelite_meteorologico)
- [Seaman,2013] SEAMAN C. *Wild Week of Wildfires, Part III*. (2013). Recuperado el 7 de Enero del 2019 de <https://podaac.jpl.nasa.gov/dataset/GOES16-OSISAF-L3C-v1.0?ids=Sensor:Platform&values=ABI:GOES-16>
- [Seaman,Miller y Hillger,2013] SEAMAN C., MILLER S. y HILLGER D. (2013). *RGB Applications of VIIRS Imagery in Support of a Weather-Ready Nation*. CoRP Symposium. Recuperado el 3 de Enero del 2019 de [http://cimss.ssec.wisc.edu/corp/2013/Tues\\_talks/2013\\_CoRP\\_Symposium\\_CIRA\\_Seaman\\_RGB.pdf](http://cimss.ssec.wisc.edu/corp/2013/Tues_talks/2013_CoRP_Symposium_CIRA_Seaman_RGB.pdf)
- [Sentinel-2-eoPortal,n.f] Sentinel-2 - eoPortal Directory - Satellite Missions.(n.f). Recuperado el 23 de Diciembre del 2018 de <https://directory.eoportal.org/web/eoportal/satellite-missions/c-missions/copernicus-sentinel-2>
- [SIGSA,2018] SIGSA. ENVI-IDL. (2018). Recuperado el 14 de Diciembre del 2018 de <http://www.sigsa.info/productos/envi/idl>
- [SMN,2017] SMN. Desarrollo. (2018). Funciones y Objetivos. Recuperado el 5 de Noviembre del 2018 de <https://smn.conagua.gob.mx/es/smn/funciones-y-objetivos>
- [Suomi-npp-eoPortal,n.f] Suomi-npp - eoPortal Directory - Satellite Missions. (n.f). Recuperado el 20 de Diciembre del 2018 de <https://directory.eoportal.org/web/eoportal/satellite-missions/s/suomi-npp>



- [Suomi\_NPP,2018] Suomi\_NPP. (2018, Septiembre 10). Recuperado el 23 de Diciembre del 2018 de [https://en.wikipedia.org/wiki/Suomi\\_NPP](https://en.wikipedia.org/wiki/Suomi_NPP)
- [Web Map Service,2019] Web Map Service. (2019). Recuperado el 16 de Enero del 2019 de [https://es.wikipedia.org/wiki/Web\\_Map\\_Service](https://es.wikipedia.org/wiki/Web_Map_Service)
- [Yu,Xu y Chen,2010] YU Y., XU H., CHEN M. y TARPLEY D. (2010). *GOES-R Advanced Baseline Imager (ABI) Algorithm Theoretical Basis Document For Land Surface Temperature*. NOAA NESDIS center for satellite applications and research. Ver. 2. 15-33. Recuperado el 29 de Diciembre del 2018 de <https://www.goes-r.gov/products/ATBDs/baseline/baseline-LST-v2.0.pdf>