Trabajo de Fin de Master.

Master en Software Libre. Universitat Oberta de Catalunya (UOC)

PAC 4. Memoria Final

Título del proyecto:

AUVReport. Software de Control y generación de informes para vehículos Autónomos Submarinos

Especialidad: Desarrollo de Aplicaciones de Software Libre.

Alumno: Pablo Rodríguez Fornes.

Consultor : Gregorio Robles Martínez

Licencia

Creative Commons

Esto es un resumen fácilmente legible del texto legal (la licencia completa).

Usted es libre de:

- **Compartir**—copiar, distribuir y comunicar públicamente la obra, y
- **Derivar**—hacer obras derivadas

Bajo las condiciones siguientes:

- **Reconocimiento**—Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciador (pero no de una manera que sugiera que tiene su apoyo o apoyan el uso que hace de su obra).
- **Compartir bajo la misma licencia**—Si transforma o modifica esta obra para crear una obra derivada, sólo puede distribuir la obra resultante bajo la misma licencia, una de similar o una de compatible.

Entendiéndose que

- **Exoneración**—Cualquiera de estas condiciones puede ser exonerada si obtiene el permiso del titular de los derechos de autor.
- **Otros derechos**—De ninguna manera son afectados por la licencia los siguientes derechos:
  - los previstos como excepciones y limitaciones de los derechos de autor, como el uso legítimo;
  - los derechos morales del autor; y
  - los derechos que otras personas puedan tener sobre la misma obra así como sobre la forma en que se utilice, tales como los derechos de imagen o de privacidad.
- **Nota**—Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra. La mejor forma para hacerlo es con un enlace a http://creativecommons.org/licenses/by-sa/3.0/deed.es

Resumen del proyecto

La presente memoria es un resumen de la actividad desarrollada para el desarrollo del Proyecto de Fin de Master de Software Libre, impartido por la Universitat Oberta de Catalunya (UOC).

El proyecto, denominado AUVReport, ha tenido tres ejes de trabajo:

- Desarrollo de herramientas e control de calidad y procesado de datos adquiridos con vehículos autónomos Submarinos gestionados por la Unidad de Tecnología Marina.
- Integración de las diferentes herramientas en un único entorno gráfico
- Generación de un informe técnico mediante el uso de las herramientas desarrolladas.

El lenguaje elegido para el desarrollo del proyecto ha sido Python, utilizando las librerías graficas Qt (v4.7) y su adaptación (binding) para Python (PyQt). El conjunto da mucha flexibilidad de utilización y potencia, una muestra de esto se ha ce patente en el hecho que la totalidad del proyecto se ha realizado en entorno Windows pero la traslación a Linux apenas necesitó la modificación de unas pocas líneas de código, realizándose en pocas horas de trabajo.

El proyecto es un punto de inicio para la realización de una aplicación más desarrollada en el mismo ámbito, con la idea de hacerla extensible a usuarios de este tipo de vehículos en instituciones oceanográficas internacionales.

CONTENIDO

1. INTRODUCCIÓN.

1.1. Motivación

Desde el punto de vista personal, la principal motivación del proyecto es el aprendizaje un lenguaje nuevo (Python en este caso) con una elevada aplicabilidad en el ámbito profesional del alumno. Para ello se ha considerado la idea de implementar una herramienta de código abierto para la gestión y control de calidad de los datos adquiridos mediante los vehículos autónomos submarinos (AUV) gestionados directamente por la Unidad de Tecnología Marina (CSIC).

En investigación oceanográfica, los datos adquiridos mediante los sensores instalados en las plataformas o desplegados desde estas, suelen ser procesados[1] con software específico del instrumento utilizado o con software comercializado por compañías especializadas (*Fledermaus*® [2], *Surfer* / *Voxler*® [3]) que permite la representación directa o programada mediante lenguajes propios de *scripting* de dichos datos.

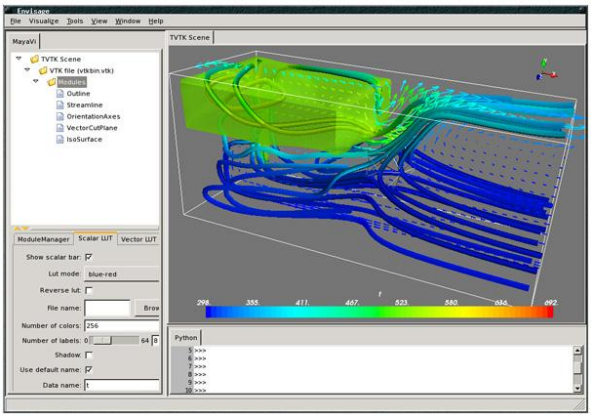


Fig. 1. Representación de datos 3D con MayaVi2

En el ámbito de la investigación es muy común que cada grupo desarrolle sus propias herramientas de procesado. Normalmente se utilizan diferentes de lenguajes (predominantemente C/C++, Fortran o Java) pero en muchas ocasiones es necesario utilizar en estos programas librerías propietarias o compiladores comerciales (*Matlab*®, NAS, ESRI, etc.).

En general, para cada una de estas aplicaciones, existe una alternativa de software de código abierto. En el ámbito científico, uno de los lenguajes que se está imponiendo por su relativa sencillez, potencia y flexibilidad es Python (en múltiples implementaciones). Este lenguaje dispone de librerías con características similares a programas comerciales equivalentes, *matplotlib*[4] en el caso de *Matlab*®, y *Blender* o *MayaVi*[5] (Fig. 1) o PyQWT[6] en el caso de representación de datos en 3D.

La utilización de software de código abierto tiene numerosas ventajas, pero en el caso del software científico algunas de las más importantes son :

- Reducción de costes de adquisición y mantenimiento de licencias.
- Mejora de la compatibilidad de las aplicaciones, tanto con nuevas versiones como con versiones antiguas (este es un problema especialmente notorio con Matlab)

1.2. El contexto tecnológico.

AUV. MERCADO ACTUAL. TIPOS Y MODELOS

Los Vehículos Autónomos Submarinos (AUV por sus siglas en inglés) son, como su nombre indica, plataformas autónomas capaces de seguir unas rutas preestablecidas y desarrollar determinadas tareas de forma autónoma. En este aspecto no difieren mucho de otros tipos de vehículos robotizados terrestres o aéreos (UAV's), aunque el control y la dinámica están adaptados lógicamente al medio marino y sus particularidades.

Históricamente los AUV's han sido un proyecto militar, sus características lo hacían una herramienta ideal para la realización de trabajos clandestinos de recopilación de inteligencia. Los primeros vehículos datan de los años 60 y eran de gran tamaño (hasta 12 metros de eslora / longitud), desde entonces esta tecnología ha estado en constante desarrollo, especialmente potenciado por la marina de guerra americana y el interés científico de grandes instituciones de investigación como MIT o Woods Hole Oceanographic Institution (WHOI) , Scripps, Acuario de Monterey (MBARI), etc.

A partir de los años 90, los avances tecnológicos y el aumento de la demanda civil (principalmente por las empresas de prospección de recursos minerales), favoreció el impulso de estas tecnología, aunque seguía siendo muy cara para el uso civil. No obstante su uso extensivo en el campo de la exploración (geofísica principalmente) marina junto con la miniaturización progresiva de plataformas y sensores su abaratamiento permitió la progresiva utilización en ámbitos no militares, tales como la supervisión de gaseoductos, búsqueda y salvamento, arqueología subacuática o monitorización ambiental) y, en los últimos años, la investigación oceanográfica en general.

Existen diferentes tipos de AUV's pero, en general, se pueden clasificar según su tamaño y capacidad de carga o *payload* (la instrumentación no incluida en el sistema de control propio del vehículo) en:

- AUV Oceánicos. Son los más grandes, miden entre 3 y 20 m. y tienen autonomías superiores a las 24 horas. La profundidad máxima que pueden alcanzar puede ser superior a 6000 m. y su capacidad de carga es superior a los 35 kg.
- AUV Ligeros. Pesan entre 50 y 300 Kg. y alcanzan profundidades de hasta 2000 m. La capacidad de carga está entorno a los 35 kg.
- Mini-AUV. Pesan menos de 50 kg y la profundidad máxima de trabajo es de 200 m. Su capacidad de carga es inferior a los 5 kg.

El modelo para el que se desarrollará el proyecto es un Oceanserver – YSI Ecomapper. Este es un vehículo ligero (apenas llega a los 40 kg), operable por una persona desde una embarcación semirrígida o incluso desde la playa.

Las principales características de este mini-auv son las siguientes:.

- Tamaño: 140 x 15 cm
- Peso (en aire): 21 kg + sensores
- Profundidad máxima de operación: 100 m (en configuración estándar) / 200 m (extendida).
- Autonomía: 8+ hrs (600 W/h batt) @ 2 nudos.

- Comunicaciones: WiFi.
- Navegación: GPS, DVL.
- Procesadores: 2 CPU's basadas en procesador Atom (uno dedicado a futuras expansiones con 8 puertos USB y 16 puertos serie), incluyendo API para interface con el procesador principal. . 2 Discos duros de estado sólido (60Gb)

Fig. 1. Vehículos listos para su despliegue.

Los vehículos tienen dos configuraciones diferentes. Uno de los vehículos tiene instalado una sonda para control de calidad de aguas, que mide diferentes parámetros físico-químicos (Conductividad, Profundidad, Temperatura, Oxígeno Disuelto (%, mg/l), clorofila, pH / ORP, turbidez, ADCP/DVL. El segundo vehículo está equipado con cámaras de video y fotografía, un perfilador de corrientes (ADCP/DVL) y un sonar de barrido lateral.

En las referencias se incluye la dirección de una página web donde se pueden encontrar todos los modelos del mercado con abundante documentación adicional.

El problema de los datos y el control del estado del AUV

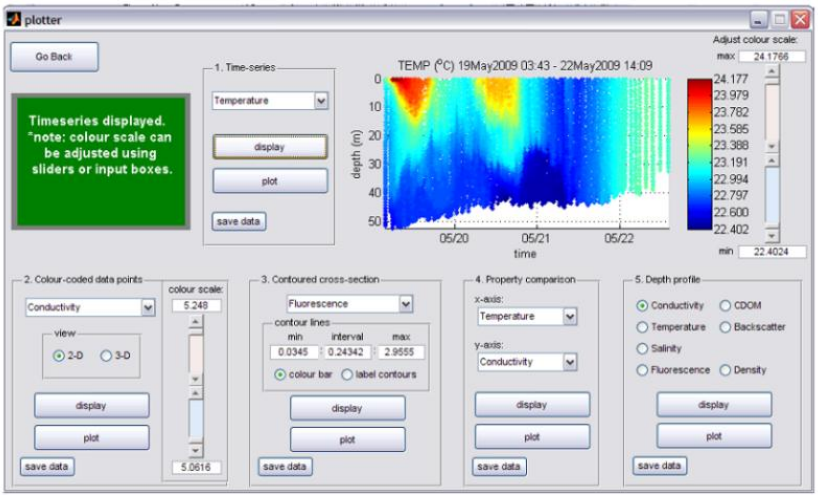
Estos vehículos generan diferentes ficheros: de datos, de variables de estado (temperatura interna, posición de los servomecanismos, etc.), alarmas. Dependiendo del fabricante estos ficheros pueden variar bastante, tanto en el formato como en la composición (variables almacenadas) que requieren un tratamiento específico para extraer la información de interés. La programación orientada a objetos se adapta muy bien a este problema.

En los anexos se han incluido muestras de dos de estos ficheros de dos vehículos diferentes.

En nuestro caso concreto, todos los datos relacionados con la dinámica del vehículo se almacenan en un solo fichero con extensión `.log`. En un principio, este es el principal fichero que se utilizará para el programa, aunque existen otros ficheros adicionales que pueden contener información adicional para futuros módulos funcionales.

En este proyecto se pretende recoger la información de ingeniería almacenada en los ficheros de estado (.log) del vehículo para generar informes científico-técnicos y los metadatos que puedan servir para mejorar la gestión de la plataforma. Podrían obtenerse, por ejemplo, datos sobre:

- Evolución de la descarga de la batería para estimación de su capacidad (y vida útil).
- Estadísticas de tiempos, profundidades y distancias de inmersión por misión para mejorar la planificación y la logística.
- Recopilación de información de misiones para realización de informes científico-técnicos.
- Estimación de dispersión de parámetros medidos para la evaluación del envejecimiento o disfunción de sensores.
- Estimación de la precisión de los sistemas de navegación.



SOFTWARE SIMILAR UTILIZADO EN EL ÁMBITO DE APLICACIÓN.

Las funcionalidades del software propuesto no suelen encontrarse en software comercial o de código abierto, pues pertenece a un ámbito bastante restringido en el que cada usuario se programa sus propias herramientas.

No obstante hay dos iniciativas interesantes muy desarrolladas que pueden servir como ejemplo en cuanto a prestaciones y objetivos generales:

GliderScope[1]. Sotware desarrollado en Australia por el IMOS (Integrated Marine Observing System) para la visualización e interpretación de datos oceanográficos adquiridos con vehículos autónomos (gliders). El software está desarrollado en Matlab por lo que es necesario disponer de este software ya instalado, aunque se distribuye libremente también con un runtime del motor Matlab. (Fig. 2)

UDel[7]: Científicos de la Universidad de New Hampshire (*CCOM/JHC*) y de la Univ. de Delaware (*Coastal Sediments, Hydrodynamics and Engineering Lab)* han desarrollado un Toolbox de Matlab para la interpretación de datos de ingeniería y científicos de un vehículo autónomo Gavia.

Fig. 2. Pantalla del programa Glidescope

1.3. Contexto Social.

El proyecto se desarrollará en la Unidad de Tecnología Marina del Consejo Superior de Investigaciones Científicas, en Barcelona. La unidad se formó en 1994 para ofrecer apoyo técnico y operacional a las operaciones en el Buque Oceanográfico Hespérides, perteneciente a la Armada Española y la Base Antártica Juan Carlos I. Desde entonces se han incorporado los buques García del Cid (1998), B/O Sarmiento de Gamboa (2007) y B/O Casitérides (2011). El año 2010 se adquirieron dos vehículos autónomos submarinos (AUV) con diferentes sensores de imagen y calidad de aguas.

El CSIC dispone de numerosas licencias de software corporativo que incluyen Sistemas Operativos (Windows), librerías matemáticas y de análisis (NAG, SPSS, MatLab, etc), compiladores (C,C++,Fortran, etc) , software de uso general (MS Office, Autocad, Windows Server, etc, Linux), GIS (ESRI). No obstante, se incrementa de forma paulatina el uso de alternativas de código abierto al software clásico licenciado.

1.4. Aplicación del proyecto.

El proyecto que se propone tiene aplicación directa en las actividades del Laboratorio de Plataformas Autónomas de la UTM. El software generado servirá para varios objetivos:

- Mejorar la gestión de la calidad de los datos adquiridos.
- Mejorar el control sobre e estado de las plataformas con un control sistemático de parámetros de funcionamiento.
- Automatizar la generación de informes científico-técnicos
- Automatizar la generación de Metadatos que cumplan con la directiva europea INSPIRE

Otros objetivos son los relacionados con la visibilidad y utilidad del proyecto:

- El software deberá ser abierto y modular para permitir adaptarlo a diferentes plataformas de características similares.
- El software se pondrá a disposición de otros usuarios para su utilización en instituciones ajenas al CSIC.
- El software estará escrito totalmente en inglés para facilitar su internacionalización

2. OBJETIVOS.

En el campo de las ciencias marinas, la mayor parte del software disponible se ocupa del análisis y visualización de los datos adquiridos con los sensores instalados en las diferentes plataformas (buques, boyas, e incluso satélites). Ya hemos visto que existen numerosos paquetes de software y librerías comerciales que se utilizan de forma habitual para desarrollar rutinas específicas para el procesado de los datos de interés para cada grupo de investigación.

Este proyecto pretende abordar una faceta que normalmente queda desatendida en estas aplicaciones, como es la visualización y análisis de parámetros relacionados directamente con el estado de las plataformas mismas (carga de baterías, estabilidad de la alimentación, etc.) y el control de la calidad de los datos adquiridos, mediante la generación automatizada de gráficos e informes.

Se pretende así mismo que el software sea modular y abierto, de modo que en un futuro pueda adaptarse de forma sencilla a diferentes tipos de vehículos. De este modo, cada usuario lo puede adaptar a las necesidades específicas de sus vehículos y ampliar las funcionalidades básicas del paquete mediante nuevos módulos. Ejemplos de nuevas funcionalidades serían la generación de metadatos compatibles con la directiva europea INSPIRE, la conversión de los datos en formatos compatibles con el estándar SeaDatanet o diferentes IDE geoespaciales (Geoservers), etc.).

La mayor parte de estos vehículos tienen software propio para el almacenamiento, análisis y representación de los datos adquiridos, normalmente desarrollado para plataformas Windows. Por este motivo el proyecto se desarrollará en una plataforma Windows aunque se pondrá especial énfasis en que todos los módulos y herramientas utilizados permitan una fácil portabilidad a Linux / OSX.

2.1. Objetivos del proyecto

El principal objetivo de este proyecto es el desarrollo de una aplicación de código abierto para la gestión y control de calidad de los vehículos autónomos submarinos (AUV) gestionados directamente por nuestra institución (CSIC), más específicamente:

- Control de calidad de datos adquiridos. Esto se hará representando gráficamente los valores de las variables medidas (temperatura, salinidad, etc.) de los ficheros de estado generados.
- Control del estado del vehículo. Representación gráfica los valores de las variables de estado (temperatura interna, estado de las baterías, comportamiento de los servos). Esto permitirá detectar anomalías en el funcionamiento del vehículo.
- Generación de estadísticas de misiones.
- Generación de informes técnicos, incluiyendo toda la información generada anteriormente.
- Desarrollo de métodos sinópticos que permitan el control de calidad de los sensores de a bordo.
- Generación de ficheros de profundidades (batimetrías) a partir de datos de correntímetro doppler, aptos para ser representados por paquetes específicos.

- Exportación de los metadatos a un formato válido para ser importado por aplicaciones de SeaDatanet, tipo *Mikado* o similar.

Otros objetivos, relacionados con la visibilidad y utilidad del proyecto son:

- El software deberá ser abierto y modular para permitir adaptarlo a diferentes plataformas de características similares.
- Esta modularidad deberá permitir el crecimiento del paquete y su adaptación a nuevas plataformas o sensores.
- El software se pondrá a disposición de otros usuarios para su utilización en instituciones ajenas al CSIC.
- El software estará escrito totalmente en inglés para facilitar su internacionalización

3. DESARROLLO.

3.1. Diseño general del programa

El diseño original del programa respondía al siguiente esquema:

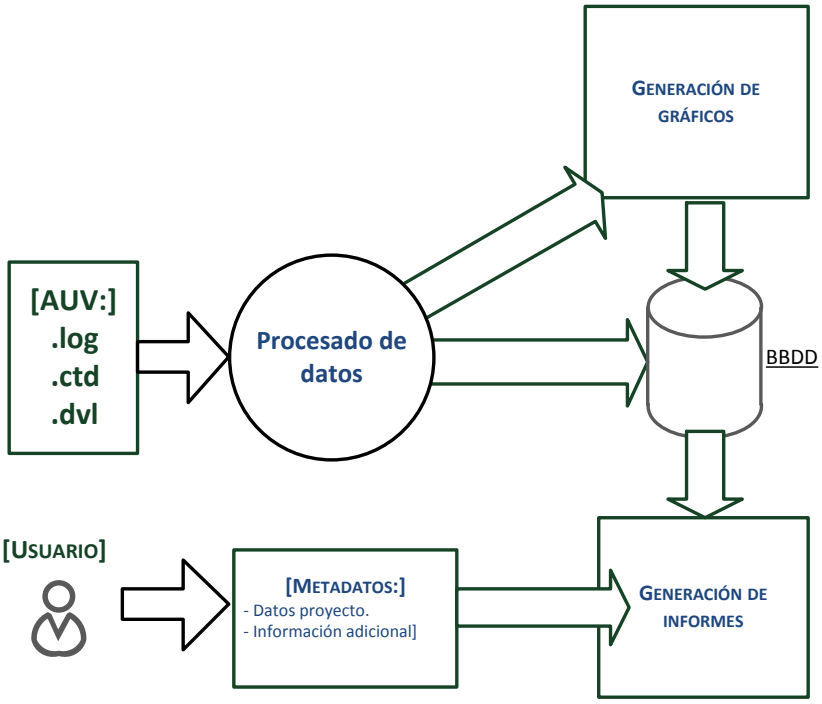


Fig. 2. Esquema de funcionamiento del programa

No obstante, en la implementación final se ha eliminado la Base de Datos, dejando los datos a representar en memoria. Para futuras aplicaciones se podría estudiar la conversión de los datos originales a un formato NetCDF para permitir su almacenamiento en Bases de Datos específicas de ámbito marino y su inclusión en Geoservidores dedicados.

Las características generales del programa serán las siguientes:

- Modularidad. Cada tarea la realizará un módulo independiente que en la medida de lo posible debería ser capaz de funcionar por si sólo o mediante llamadas parametrizadas desde la consola.
- Posibilidades de expansión. El programa deberá ser fácilmente expandible, tanto el número de módulos como en las funcionalidades de los módulos.
- Multivehículo. El programa deberá ser adaptable a diferentes vehículos de diferentes fabricantes.

En principio, la base de datos y los ficheros (o directorios) de datos estarán o en la máquina local o en una unidad de red. No se realizará ningún tipo de acceso vía web.

3.2. Herramientas.

Las herramientas utilizadas en el desarrollo del proyecto han sido las siguientes:

- Para la evaluación del código se utilizarán las utilidades PEP8, PyLint (Integrado en el entorno Eric) y el generador de estadísticas de métrica integrado también en el propio entorno.
- Para la generación de documentación y referencia se ha utilizado Epydoc[8]

También se han utilizado algunos módulos para tareas específicas (como la conversión de coordenadas) disponibles en la red bajo diferentes licencias OS.

SISTEMA OPERATIVO

El proyecto se ha desarrollado en Windows 7 pero también se ha probado en una maquina virtual con Fedora16.

PYTHON

El lenguaje elegido para la realización del proyecto ha sido Python[9] en su versión 2.7.2.

Python es un lenguaje *interpretado* de código abierto multiplataforma y de alto nivel, creado a finales de los años 80 por Guido Van Rossum. Python soporta diferentes paradigmas de programación, incluyendo la programación orientada a objetos, y tiene numerosas implementaciones (*bindings*) en lenguajes como C (*CPython*), C++, Java (*Jython*) o .Net (*IronPython*), entre otros.

Una muestra de su popularidad, la encontramos en el índice TIOBE de mayo de 2012 (http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html) que lo posiciona como uno de los 10 lenguajes más utilizados.

La modularidad es una característica muy importante de Python. El código escrito en Python puede organizarse en módulos independientes que contienen funciones o clases que proporcionan funcionalidades adicionales a las de los módulos de las distribuciones básicas. Además el usuario puede organizar sus propios módulos para reutilizar de forma flexible el código generado.

Actualmente existen dos distribuciones: Python 2.X y Python 3.X, las diferencias entre ambas ramas son básicamente sintácticas y están detalladas aquí[10]. No obstante existe un pequeño problema en esta duplicidad de distribuciones, y es que muchos módulos (Matplotlib, por ejemplo) están desarrollados para la rama de Python 2.X y no para la más avanzada 3.X. Por este motivo, la distribución elegida para el proyecto ha sido la 2.7.X, lo que ha influenciado la elección de alguno de la los módulos utilizados. El desarrollo de la rama 2.7 está parado y se espera que la mayoría de los módulos oficiales acaben migrando paulatinamente a la nueva distribución.

Al ser un lenguaje interpretado, Python es relativamente lento (comparado con C por ejemplo), a pesar de esto, su gran flexibilidad y una amplia comunidad de usuarios lo ha convertido en un lenguaje extremadamente popular en el ámbito científico, utilizándose en campos tan diferentes como las IDE Geoespaciales (Grass / QGis) o el análisis de imágenes astrofísicas o de medicina.

En el aspecto gráfico, Python ofrece diferentes posibilidades de librerías de desarrollo (Tck-TK, Wx, Gtk, Qt) pero se ha optado por utilizar la implementación en python de las librerías Qt (PyQT v4.7.2) desarrolladas por la empresa Riverbank y que se distribuyen bajo tres licencias, una comercial y dos libres (GPL v.2 / v.3). Para el diseño del entorno gráfico en sí mismo se ha utilizado QtDesigner.

LIBRERÍAS .

Python es un lenguaje general de programación y scripting, para realizar tareas más complejas se han desarrollados módulos específicos que se pueden enlazar con el código para dotarlo de nuevas prestaciones o mejorar otras. Las librerías adicionales más importantes utilizadas en el proyecto han sido:

- Matplotlib [4]. Módulo para de representación gráfica de variables en 2D y 3D. Esta librería exige la instalación previa de librerías de cálculo numérico y científico (Numpy) que facilita el optimiza de datos en forma de matrices y vectores de una forma muy similar a Matlab.
- ReportLab.[11]. La empresa ReportLab ha desarrollado diferentes herramientas de generación de documentos y ha publicado parte de las librerías utilizadas como Open Source. ReportLab es básicamente un conjunto de librerías que permiten la generación de documentos `pdf` de manera programática utilizando diferentes metodologías. Las librerías tienen una gran flexibilidad en cuanto a los tipos de letra, y posibilidades de maquetación de los documentos.

ENTORNO DE DESARROLLO

Para el desarrollo del proyecto se ha elegido el sistema de desarrollo Eric[12] (versión 4.4.7). Entorno de desarrollo multiplataforma basado en PyQt y que integra todas las herramientas necesarias para el desarrollo del proyecto en un único entorno gráfico, como pueden ser un cliente de Subversión o Mercurial, enlaces directos a QtDesigner(Editor de ventanas Qt), control de sintaxis y autocompletado, depuración integrada, cálculo de métricas y profiling del código, etc.

Existe también una versión (Eric5) para Python 3.X

Este entorno permite la inclusión de diferentes *plugins* por lo que se le pueden añadir diferentes funcionalidades como la generación de ejecutables (cxFreeze).

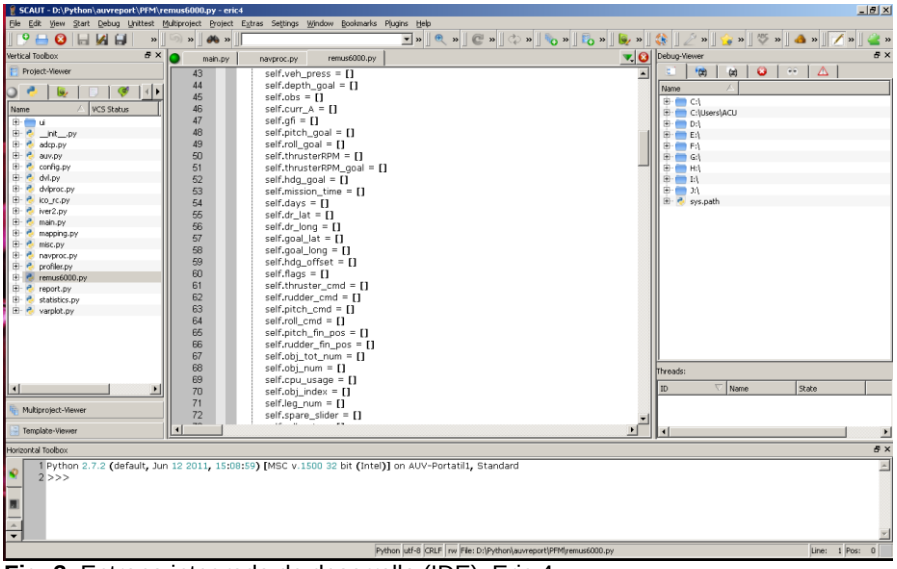


Fig. 3. Entrono integrado de desarrollo (IDE) Eric 4

SISTEMA DE CONTROL DE VERSIONES (CVS)

El entorno Eric permite la utilización integrada con repositorios SVN. En este caso se ha utilizado un repositorio público de Rediris.

(https://forja.rediris.es/projects/auvreport/).

Este repositorio dispone de todos los servicios necesarios para el desarrollo y administración del proyecto (sistema de seguimiento de errores, gestión de tareas, listas de correo, repositorio SVN, etc).

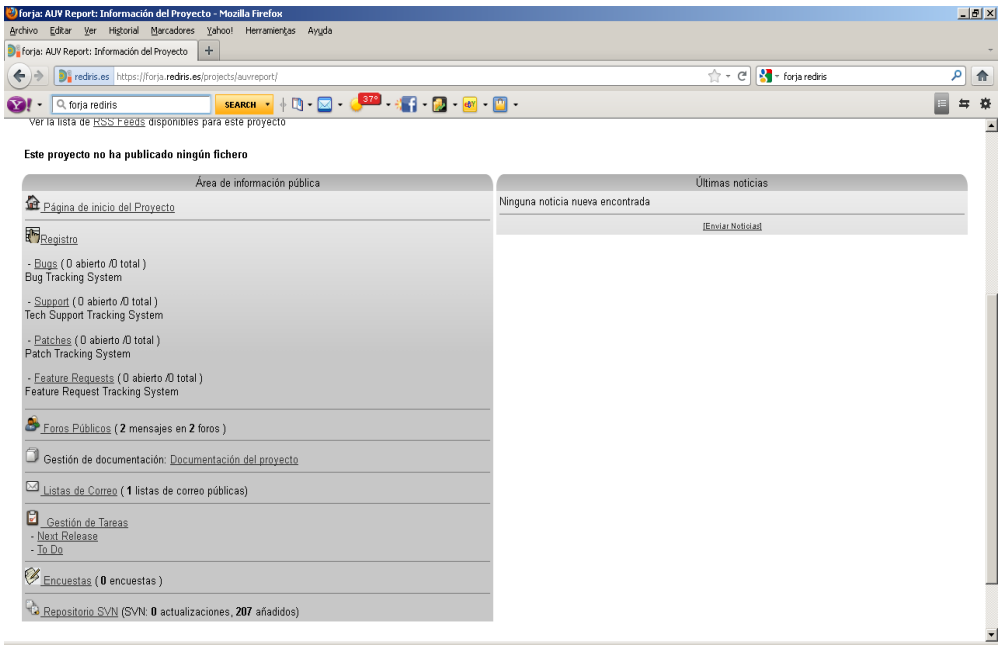


Fig. 4. Página principal del proyecto en la forja RedIris

Para tener acceso a todas las funcionalidades es necesario registrarse, pero es posible hacer un *checkout* completo del código como usuario anónimo:

```
svn checkout https://forja.rediris.es/svn/auvreport
```

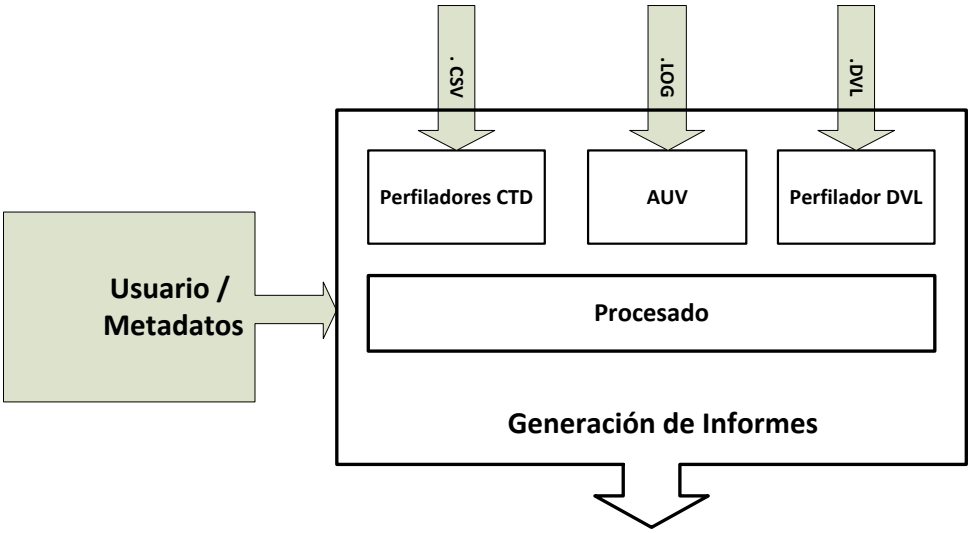
DOCUMENTACIÓN Y AYUDA EN LÍNEA

Este es un proyecto activo y dinámico, la intención inicial es modificar el código y ampliar sus funcionalidades por lo que una buena documentación de las funciones es importante. En este sentido se ha utilizado la herramienta `Epydoc`, que mediante la inclusión de metacaracteres en un *docstring* permite la generación de un completo sistema de referencia en línea, basado en documentos HTML. La inclusión de esta referencia en el código del programa es muy sencilla es accesible desde el menú principal.

Por otra parte también se ha hecho un pequeño manual de usuario en formato `html` que también es accesible desde el menú principal y que se ha incluido en los anexos como referencia.

3.3. Estructura del programa

Inicialmente el programa se concibió para poner en un mismo entorno diferentes utilidades que realizaban tareas específicas sobre los ficheros de estado de los vehículos, poniéndolas en un entorno común y aprovechando esta circunstancia para generar un informe con la información generada por cada uno de los módulos.



Este hecho ha restringido parte del diseño y presenta ventajas e inconvenientes. La principal ventaja está en el desarrollo inicial, pues cada uno de los módulos se ha podido programar y probar por separado. Por el contrario se pierde algo de eficiencia, especialmente en el acceso a disco pues existen algunos procesos redundantes que es necesario optimizar.

Se incluye a continuación un diagrama de las principales clases utilizadas.

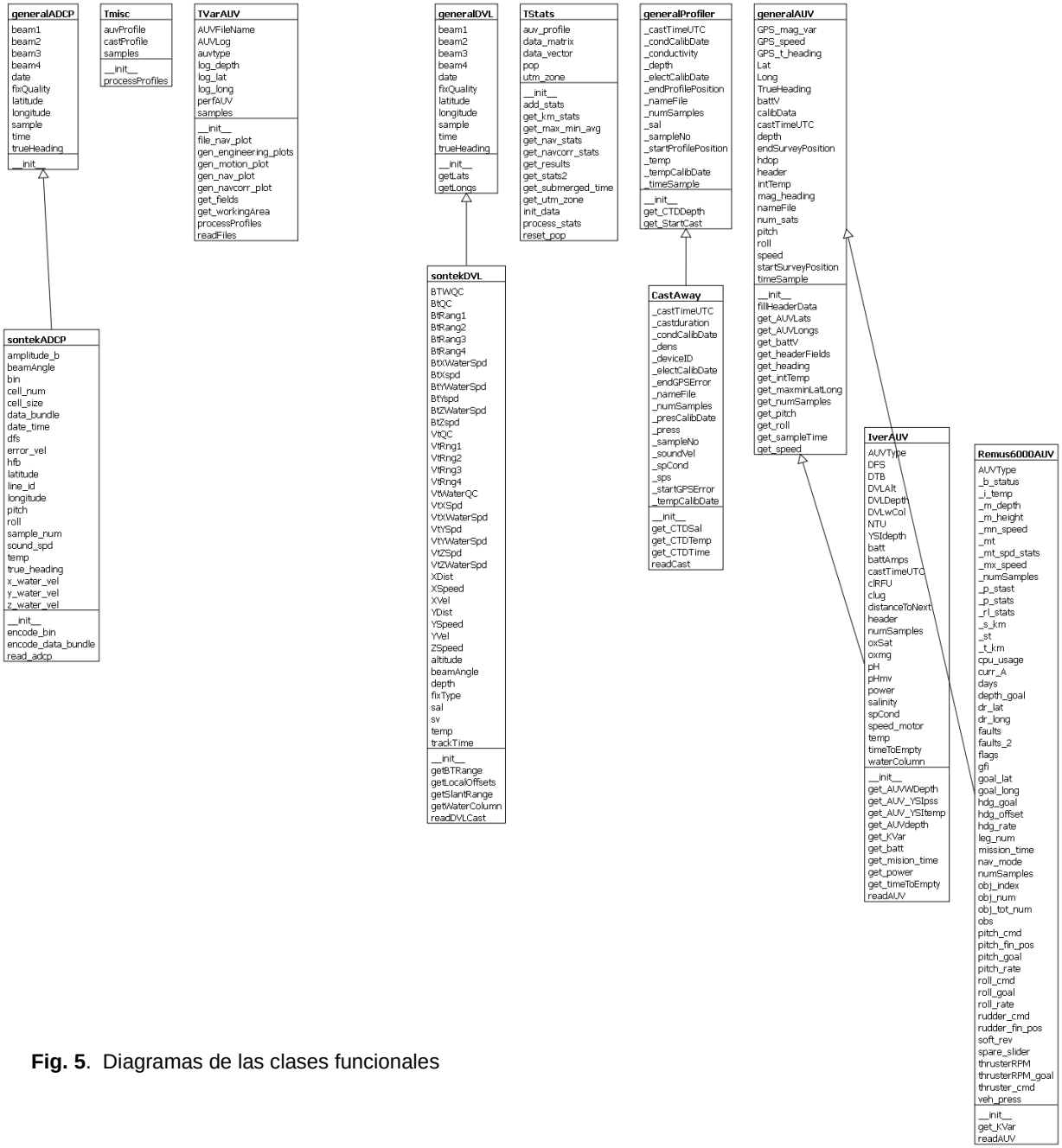


Fig. 5. Diagramas de las clases funcionales

Para tratar los datos contenidos en cada tipo de fichero se ha creado una clase por tipo de dato que incluye los métodos para su tratamiento así como métodos para hacer públicos sus datos a métodos de otros objetos.

Esto permitirá ampliar las funcionalidades en un futuro, por ejemplo, ampliando el número de modelos de AUV que el programa sea capaz de interpretar o diferentes sensores a los incluidos ahora, desarrollando las clases (heredadas o nuevas) necesarias.

A parte de las clases puramente funcionales mostradas en la Fig. 5, cada una de las ventanas diseñadas se sustenta en una clase independiente, que a su vez es heredado por una clase superior que agrupa todas las funcionalidades propias de la ventana (eventos, selecciones, etc)

La otra entrada de datos proviene del usuario, que debe introducir una serie de metadatos de forma manual a través de una ventana con pestañas. En función de las opciones seleccionadas por el usuario, el programa selecciona las opciones de procesado para cada tipo de dato y genera el informe.

3.4. Planificación.

La planificación que se ha seguido en el proyecto corresponde a la siguiente distribución

Semestre 1

1. **Planificación y diseño.**
   Definición formal del alcance del programa, sus módulos y salidas esperadas
   1.1. Definición de módulos
   1.2. Recopilación de información (formatos de datos de otros vehículos)
   1.3. Definición de formatos.
2. **Codificación:**
   2.1. Diseño del entorno
   2.2. Prototipos de módulos.
   2.3. Gestión de Entradas

Semestre 2:

3. **Planificación y diseño.**
   Definición formal del alcance del programa, sus módulos y salidas esperadas
   3.1. Definición de módulos (reporting, configuración de sensores)
   3.2. Definición de formatos.
4. **Codificación:**
   - Integración de los diferentes módulos.
   - Generación de informes y documentos de salida.

5. **Pruebas y documentación**
6. **Despliegue.**
   - Generación de ejecutables y migración a Linux
   - Informe final y presentación.

3.5. Métrica

El entorno de desarrollo dispone de una aplicación integrada para mostrar algunos parámetros relacionados con la métrica del proyecto, que en este caso se resumen en la siguiente figura:

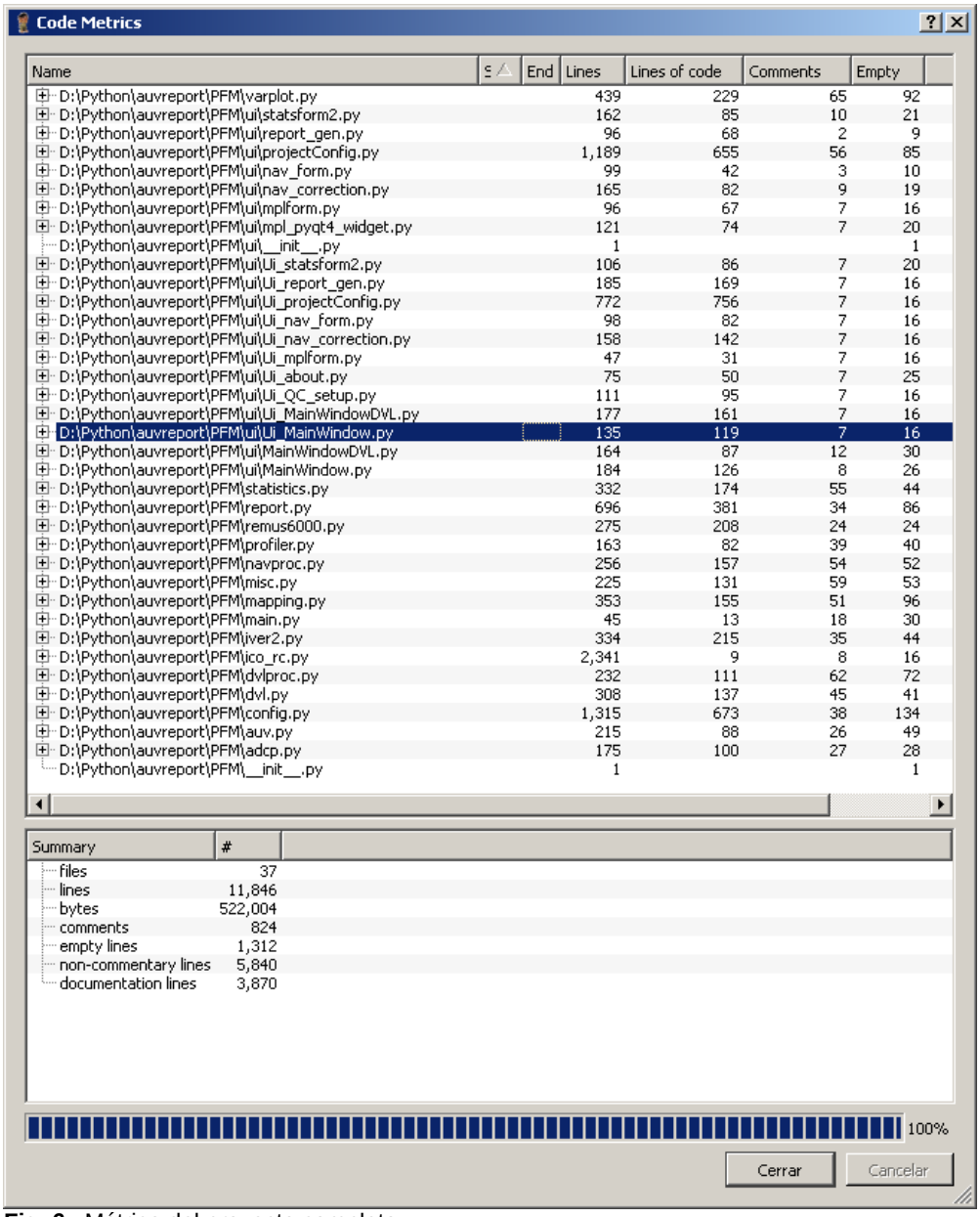


Fig. 6. Métrica del proyecto completo.

4. RESULTADOS.

4.1. Requerimientos e instalación

Se ha incluido un fichero INSTALL.txt con las instrucciones de instalación y los requerimientos para que el software funcione correctamente.

REQUERIMIENTOS:

Para el correcto funcionamiento del programa son necesarios los siguientes paquetes:

- Python 2.7.X. [9]. Distribución básica de Python.
- Numpy [13]. Librería de cálculo numeric, es necesaria para Matplotlib.
- Matplotlib [4]. Librerías de representación de datos con características similares a Matlab
- Basemap. Extensión de Matplotlib para representación de información georreferenciada. (http://matplotlib.github.com/basemap/users/download.html)
- ReportLab[14] : librerías para generación de documentos pdf de forma programática, incuyendo las librerías PIL para manipulación de imágenes.

INSTALACIÓN

La instalación del programa es sencilla:

- Instalar los paquetes según el orden indicado anteriormente.
- Descargar los ficheros desde el repositorio

```
svn checkout https://forja.rediris.es/svn/auvreport
```

EJECUCIÓN

El programa se ejecuta con el siguiente comando desde una consola

```
$> python main.py
```

4.2. Aspectos generales del programa

Al iniciar el programa aparece un pequeño menú flotante con las principales opciones

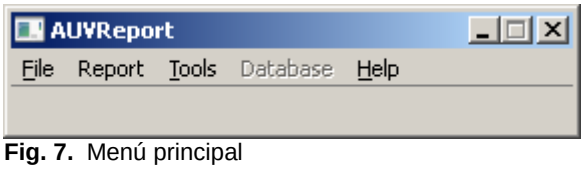


Fig. 7. Menú principal

ENTRADA DE DATOS

Como se ha comentado anteriormente, el programa tiene dos vías de entrada de datos diferentes:

En primer lugar, el usuario deberá rellenar una ficha de campaña (ver fig. 8), accesible a través del menú `FILE/New`, o `File/Edit` si se desea modificar una configuración ya existente. En estas pestañas se introducirá información sobre la ubicación de las fuentes de datos y directorios de destino e información descriptiva del proyecto (metadatos).

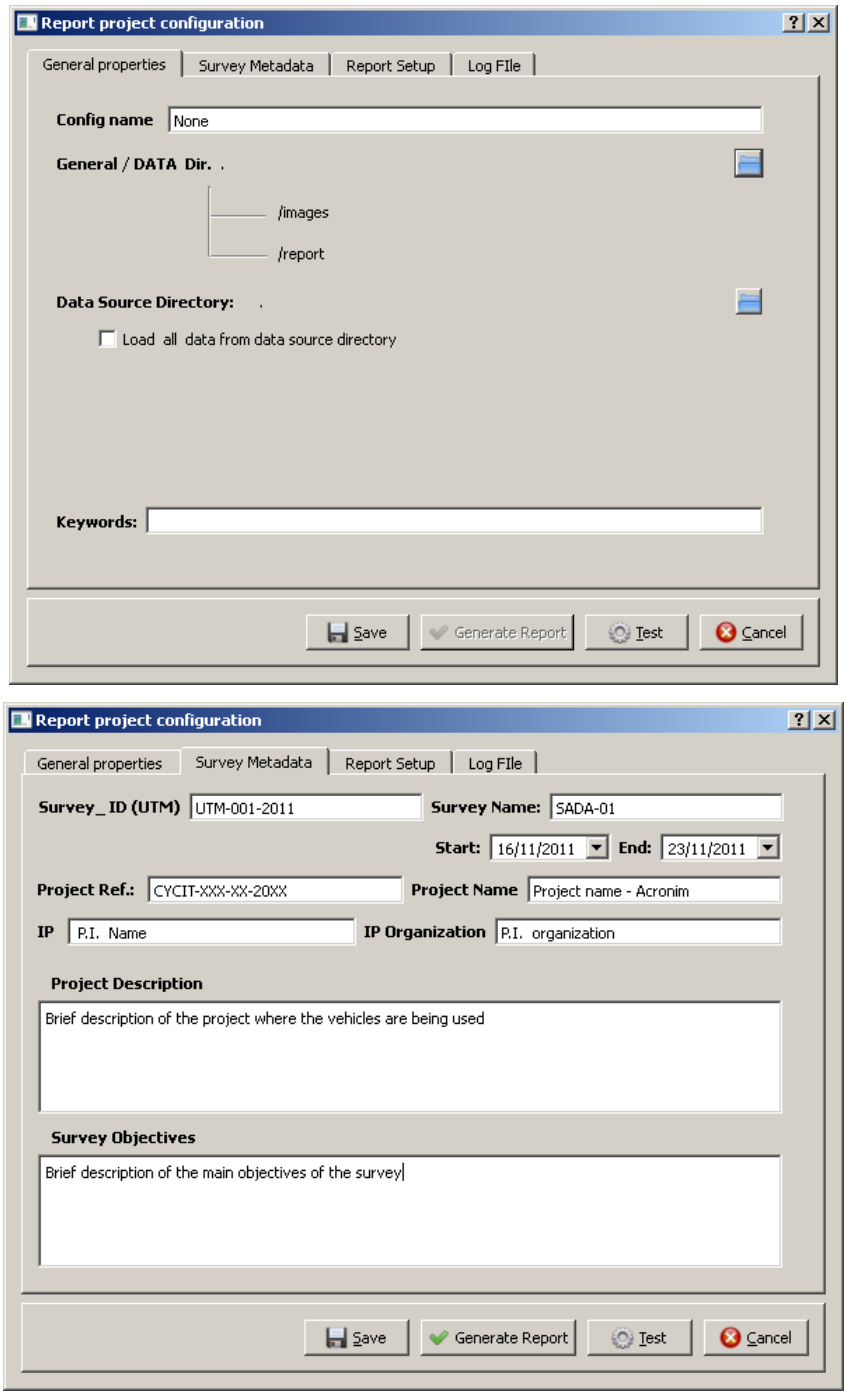


Fig. 8. Introducción de metadatos

Una vez introducida la información del proyecto, en la pestaña “Log” se podrá seleccionar el fichero de estado cuyos datos se deseen visualizar. En el caso particular de nuestro vehículo (Iver2), este fichero contiene tanto los datos de “ingeniería” como datos de la instrumentación asociada.

Si se desa incluir en el informe todos los ficheros de un determinado directorio hay que activar la casilla correspondiente debajo de la etiqueta “Data Source Directory”

Todos estos datos, incluyendo los metadatos del despliegue (o campaña) introducidos por el usuario se utilizarán posteriormente para la generación de informes científico-técnicos.

La configuración podrá guardarse en un fichero XML presionando el botón “Save”

Como se ha comentado anteriormente, el programa deberá ser capaz de leer diferentes tipos de ficheros de datos, característicos de cada modelo de vehículo (ver anexo), para esto será necesario que la primera línea del fichero contenga el nombre de las variables a considerar, que se almacenarán en una lista que será la que utilice el programa para determinar qué variables se graficarán y como.

Hay vehículos en los que los ficheros de estado o datos no están organizados de forma tabular, en principio esta posibilidad no se contempla pero debería ser fácilmente convertible a un formato tabular mediante un script intermedio.

A partir de este fichero se podrán generar (botón Test) gráficas representativas de los parámetros de interés que se podrán almacenar como ficheros gráficos (formato `.png`) para su posterior inclusión en el informe técnico u otros informes.

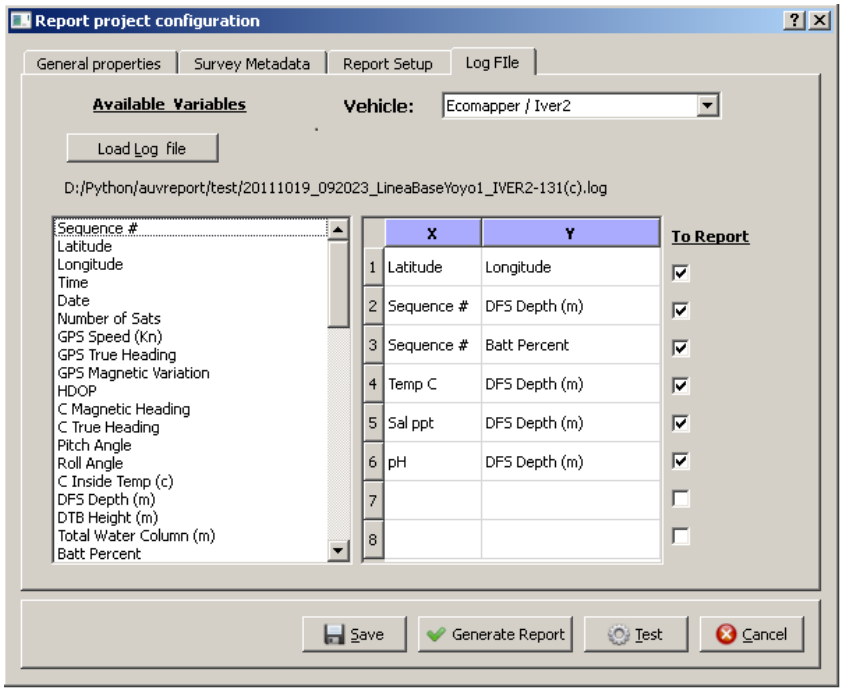


Fig. 9. Interfase de entrada de datos desde los ficheros de estado

PROCESADO / HERRAMIENTAS

El programa está organizado en diferentes módulos, cada uno de los cuales realiza un tipo de procesado diferente, siendo la interfase de usuario la responsable de seleccionar los ficheros a procesar en cada caso. Estos módulos son accesibles individualmente desde el menú **`Tools`** de la ventana principal

Se podrán añadir nuevas funcionalidades (en forma de módulos adicionales) en un futuro.

En concreto, los módulos que se han implementado en este proyecto serán los siguientes.

Control de Calidad de datos:

Un problema fundamental en la adquisición de datos oceanográficos es el control de calidad.

Normalmente los instrumentos se calibran de forma regular en laboratorios especializados o mediante protocolos establecidos por los fabricantes de los diferentes sensores a utilizar. No obstante, a parte de esta calibración instrumental, se suele

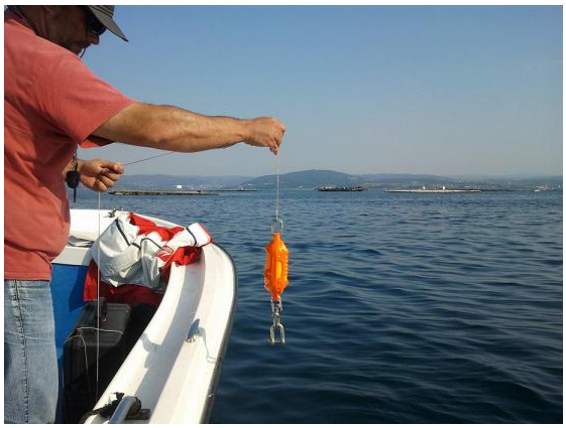


Fig. 10. Inicio de un perfil CTD

realizar una calibración cruzada con otros métodos de medición independientes para comprobar en el campo la precisión del sensor y corregir posibles derivas instrumentales o detectar problemas en los sensores.

Para ello, mientras el vehículo sigue una trayectoria previamente programada se realizan perfiles verticales con un perfilador CTD (Fig. 5) que básicamente es un sensor de profundidad, temperatura y conductividad. Este sensor tiene una precisión 3 veces superior a la de los instrumentos embarcados en el vehículo y aunque no tiene precisión suficiente para realizar una calibración instrumental, sí que servirá para realizar comprobaciones in-situ de la calidad de los datos y detectar posibles anomalías en el funcionamiento de los sensores del vehículo mediante la comparación cruzada de parámetros físico-químicos de los sensores del AUV y del perfilador CTD realizados en las proximidades de la trayectoria del vehículo permitirán realizar un primer control de calidad comprobando si existen desviaciones sistemáticas de consideración entre ambos sensores lo que podría ser indicativo de un fallo en los sensores o su calibración.

Para comparar estos parámetros (temperatura y salinidad) se generarán histogramas (Fig. 11.) donde se pueden observar las distribuciones de las diferencias de los valores de los parámetros entre ambos instrumentos en aquellos puntos donde las trayectorias de ambos son próximas (Fig. 12. En este caso la trayectoria azul es la del vehículo y los perfil de colores los CTD)

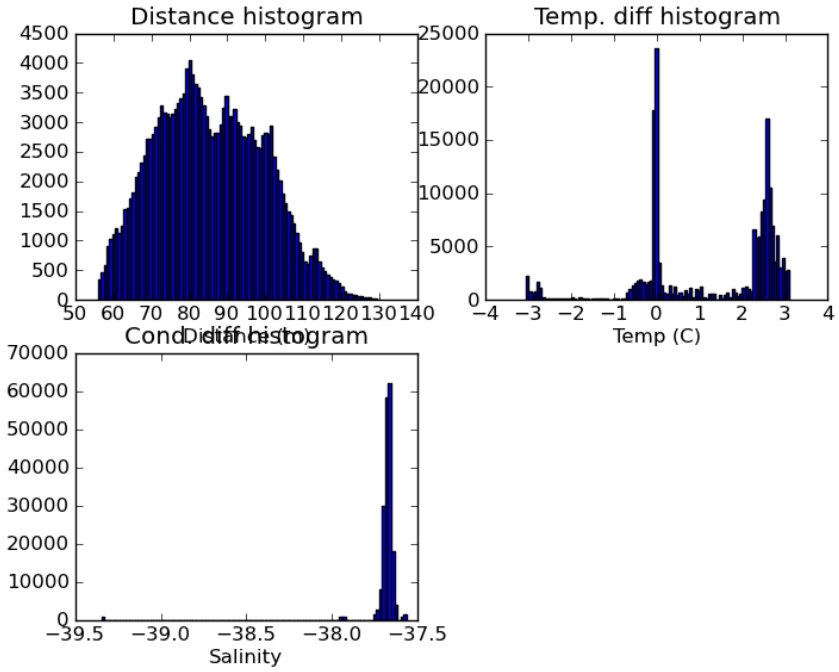


Fig. 11. Histogramas de control de calidad

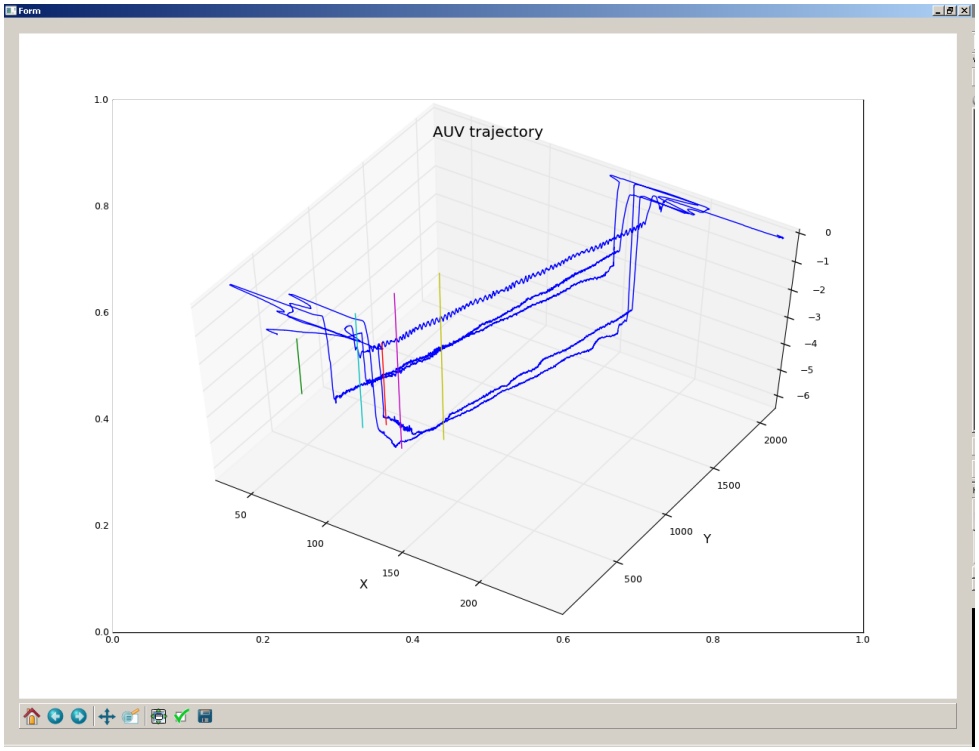


Fig. 12. Trayectoria del AUV y perfiles CTD

En este caso particular se observa:

- La mayor parte de los puntos usados como referencia estaban separados entre 50 y 120 m. de distancia.
- Se observa poca dispersión de la salinidad.
- Se observa un pico de diferencias de temperatura importante (de 2 a 3 grados) probablemente debido a que parte de la trayectoria se realizó en superficie y otra parte sumergido. Lo puntos con diferencias negativas pueden ser debidos a medidas erróneas debido a la presencia de aire en el sensor mientras el vehículo navegaba en superficie.

Graficado de Parametros internos (fichero `.log`)

Se graficarán las diferentes variables, tal como fueron seleccionadas en la pestaña de selección de datos (ver Fig. 9). Estas gráficas mostrarán el estado del vehículo o de cualquiera de los sensores cuyas medidas estén reflejadas en este fichero de estado y permitirán, por ejemplo monitorizar la temperatura interna. Si se activa la casilla del extremo derecho esta gráfica se almacenará en el disco para su inclusión posterior en el informe.

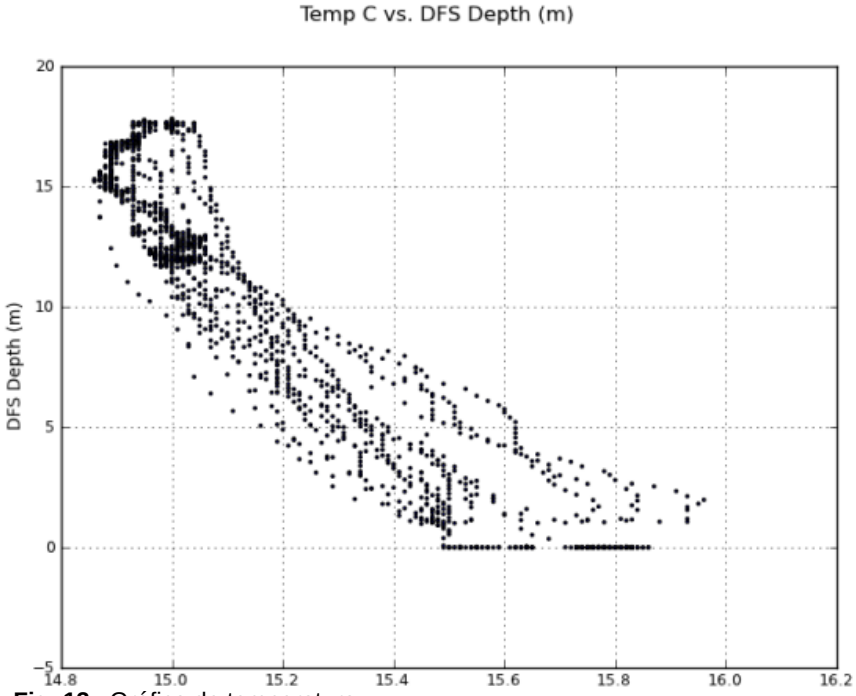


Fig. 13. Gráfica de temperatura

Generación de batimetría a partir de perfilador de velocidad (DVL):

Este tipo de vehículos suele estar dotado de un correntímetro acústico de efecto doppler (ADCP / DVL) que básicamente sirve para cuantificar (en módulo y dirección) la intensidad de la corriente marina en la columna de agua inmediatamente debajo del vehículo.

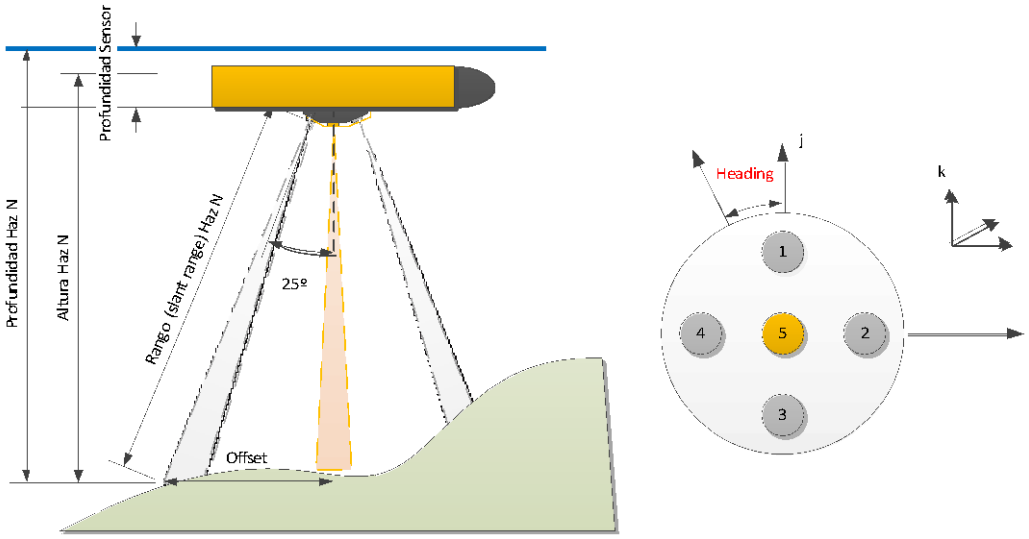


Fig. 14. Geometría de transmisión DVL y altímetro (amarillo).

No obstante, si el vehículo se encuentra lo suficientemente cerca (en nuestro caso a menos de 40 m.) del fondo marino, es capaz de recibir reflexiones del fondo marino y determinar su altura, velocidad y rumbo de traslación respecto a este. A partir de aquí, si se conoce la profundidad del vehículo y su actitud (cabeceo, balanceo y rumbo) se puede determinar la posición y profundidad (XYZ) de los "puntos de impacto" de cada uno de los 4 haces del correntímetro, obteniéndose información más completa sobre la morfología del fondo marino que con una simple sonda monohaz.

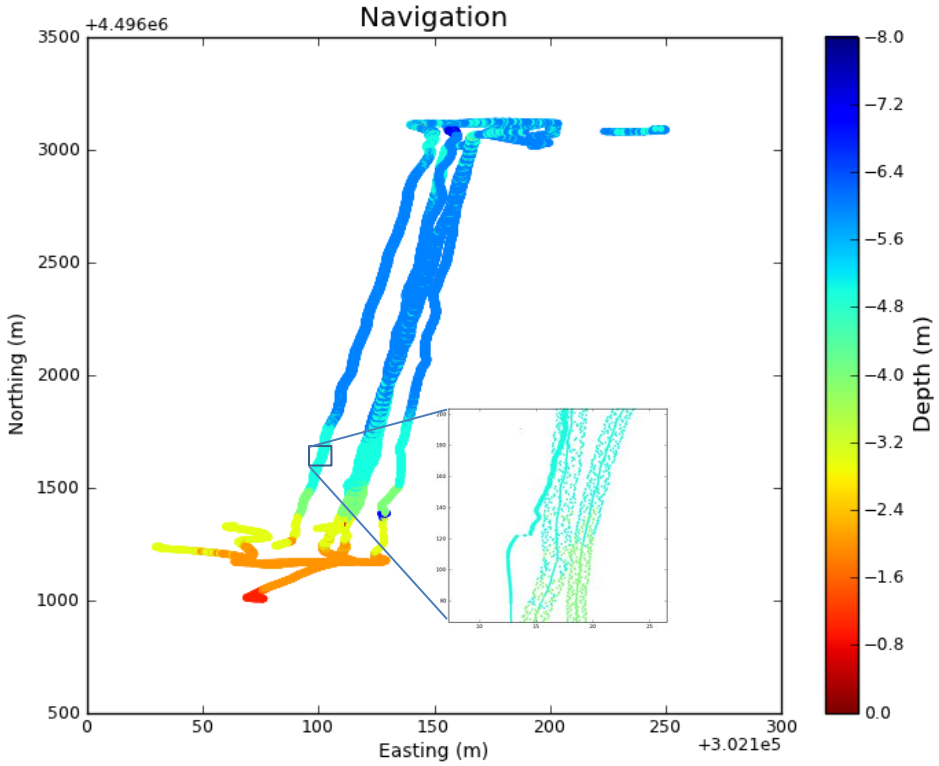


Fig. 15. Generación de datos batimétricos.

De este modo, planificando adecuadamente las trayectorias del vehículo es posible obtener una conjunto de datos batimétricos homogéneo, lo que mejora notablemente la calidad de los mapas de fondo que se pueden obtener.

Corrección de trayectorias:

Este tipo de vehículos dispone de receptores GPS que, mientras se encuentran en superficie, son capaces de determinar su posición con una precisión inferior a los 3 metros (en general), no obstante, una vez sumergidos no hay posibilidad de recibir señal GPS y es necesario acudir a otros sistemas de guiado.

En concreto, este vehículo dispone de un compás magnético y un correntimetro para realizar una navegación a estima entre dos puntos cualesquiera de la trayectoria planificada (línea roja en la Fig. 16). No obstante, una vez sumergido el vehículo puede desviarse (línea negra en Fig. 16) de su ruta programada. Una vez de nuevo en superficie el vehículo regresa a la ruta original para buscar el punto final programado. El resultado es que existe una diferencia entre la ruta programada y la que realmente ha seguido el vehículo (Fig. 16 y 17) que es necesario corregir para georreferenciar correctamente los datos obtenidos.

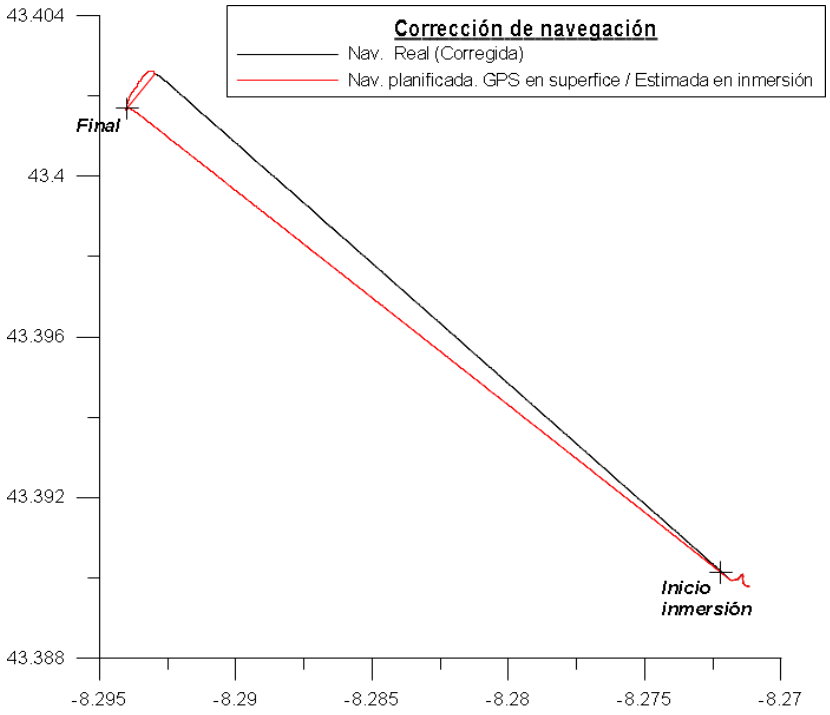


Fig. 16. Trayectoria planificada

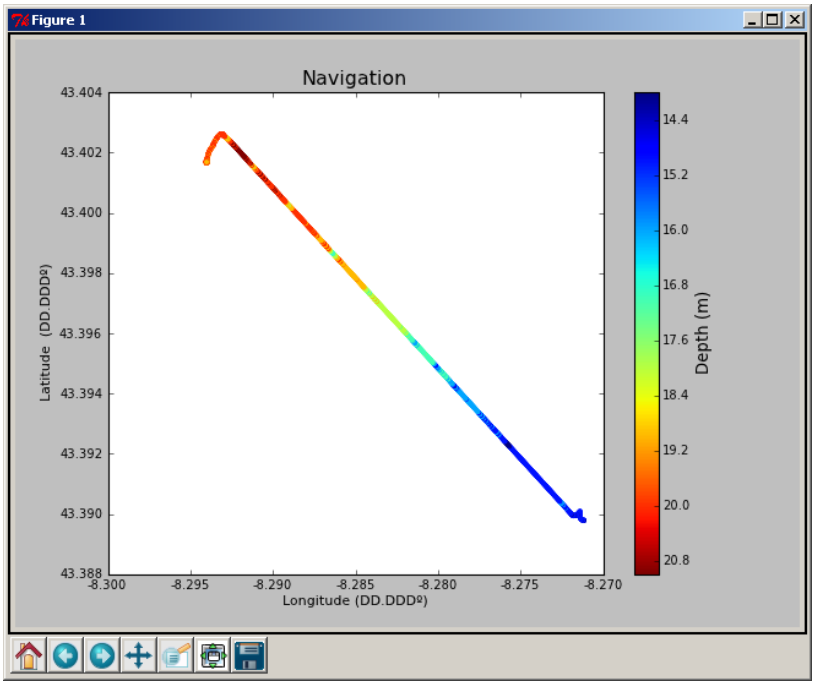


Fig. 17. Trayectoria real con profundidades

SALIDA DE DATOS

La salida de datos se realizará de dos formas diferentes:

- Gráfica. Serán las gráficas que el usuario podrá ir viendo conforme manipule el programa. Alguna de estas gráficas podrán guardarse como ficheros gráficos para usos posteriores.
- Informes. A partir de los metadatos y de los indicadores almacenados se generará un pequeño informe científico-técnico en formato `pdf` con toda la información seleccionada.

4.3. Problemas encontrados.

Los principales problemas encontrados han sido los siguientes:

- Integración de los gráficos de matPlotlib en ventanas emergentes. Matplotlib dispone de su propia interface gráfica pero se han integrado estas gráficas dentro del entorno gráfico de PyQt. Para ello se desarrollado una subclase "lienzo" a partir de la clase *FigureCanvasQtAgg*, de las librerías gráficas de matplotlib para este entorno.[15]
  El principal problema que existía aquí era mantener el control sobre las ventanas cuando se solicitan diferentes plots y se abren sucesivas ventanas modales. Para solucionar este problema, lo que se ha hecho es generar una tupla que mantiene referencias a las ventanas abiertas y actualizarla cada vez que se añade o desaparece una ventana. Falta por solucionar el cierre automático de todas las ventanas cuando se cierra la ventana madre.

- Adaptación del código para la ejecución en Linux. A pesar de que todas las librerías utilizadas son completamente multiplataforma ha sido necesario realizar un par de pequeñas modificaciones en el código para adaptarlo a las particularidades del sistema operativo.
  La manera más sencilla es utilizar funciones de la librería os que permiten acceso directo a funciones del sistema operativo. Por ejemplo, mediante la función `os.name` se puede averiguar el S.O. y a partir de ahí ejecutar el código apropiado
- Intercambio de información entre ventanas. Este problema es debido a la compartimentación del código. El lugar más evidente es en la generación del informe, ya que mientras se está generando el informe no se puede enviar información a la ventana de generación de informes para actualizar los resultados.
  Se podría solucionar esto incluyendo la generación de informes como una subclase de la ventana, mejorando la visibilidad de las variables.
- Optimización del código. El código tiene algunos cuellos de botella que es necesario eliminar mediante la optimización del código. El hecho que los módulos se hayan diseñado para realizar tareas independientes dificulta en ocasiones el traspaso de información entre ellos ejecutándose algunos métodos de forma redundante.

4.4. Desarrollo Futuro.

El programa funciona de forma estable, tanto en Windows7 como en Linux (Fedora16), aunque ni mucho menos se puede considerar un proyecto cerrado pues algunas de las características originales no se han podido implementar todavía y muchas de las implementadas tienen margen para la mejora.

Esta previsto utilizar el repositorio ya establecido como una base para un proyecto interno de la Unidad de Tecnología Marina con vistas a generar un programa mucho más completo que el presentado aquí.

MEJORAS.

Las principales mejoras que están previstas para el futuro inmediato son:

- Reimplementar la jerarquía de clases con los siguientes objetivos:
  - Permitir una comunicación fluida entre procesos y ventanas.
  - Optimizar los procesos de lectura y escritura de datos de modo que cada fichero se lea una única vez. Esto puede crear problemas si se leen muchos ficheros o se aplican diferentes tipos de procesado a una parte de los ficheros seleccionados originariamente.
  - Facilitar la incorporación de nuevos algoritmos / scripts de procesado.
  - Flexibilizar la generación del informe.
- Incorporación de nuevos gráficos con gridding de datos para generar informes científicos preliminares
- Exportación a ficheros (XML) de los resultados de estadísticas.
- Generación de ficheros compatibles con la directiva INSPIRE

- Exportación de datos a formato NetCDF con vista a su integración en un GIS marino

5. CONCLUSIONES

Se presentan a continuación las principales conclusiones en cuanto al presente proyecto:

- Python un lenguaje con una aplicabilidad muy interesante en el ámbito del software científico y para la gestión y procesado de datos. Por lo que he podido observar durante la realización del proyecto creo podría sustituir sin grandes problemas paquetes de software propietario de uso corporativo.
- Existe una enorme comunidad de usuarios en internet con mucha información disponible para los usuarios.
- Se han probado diferentes entornos de desarrollo (Eric, Spyder-Pythonxy). El primero está mas indicado a la realización de proyectos completos y presenta dos versiones diferentes para Python 2.x y Python 3.X.
  PythonXY es mas cómodo para la realización de scripts simples y para personas acostumbradas a trabajar con Matlab. En Linux requiere la instalación de un debugger separado.
- El software es operativo, aunque necesita optimizar aquellos métodos de intercambio de información y de acceso al disco.

5.1. Aplicabilidad de lo aprendido durante el Master:

En este proyecto se han aplicado muchos los aspectos aprendidos durante el Master de Software Libre, siendo los más importantes:

- La programación del proyecto. Al compaginar la realización del proyecto con el trabajo habitual el periodo de programación ha sido extenso pero se ha intentado seguir las directrices aprendidas durante el curso en la medida de lo posible, en especial lo relacionado con la temporalización de tareas y la asignación de recursos (temporales fundamentalmente)
- Uso de lenguajes y herramientas libres. Este aspecto tiene una especial relevancia para mí por la aplicabilidad inmediata a mi entorno laboral paa la realización de nuevos proyectos.
- Uso de herramientas para documentación y depuración del código (Epydoc, Plynt, pep8), han permitido mejorar un poco la legibilidad del código y su adecuación a los estándares de programación, aunque su utilización no ha sido muy intensiva (y tardía en cuanto al desarrollo del proyecto), ha permitido entrar en contacto con estas herramientas y aprender los beneficios de su utilización desde el inicio del proyecto.
- Programación de entornos gráficos multiplataforma.
- Utilización de plataformas colaborativas para el desarrollo de proyectos.

Por supuesto, el proyecto no se acaba aquí, hay muchos aspectos a mejorar, pero desde el punto de vista personal representa un muy interesante punto de inicio para futuros proyectos.

Pablo Rodríguez Fornes

Barcelona, 7 de junio de 2012

6. ANEXOS

6.1. Formato del fichero de datos (.log)

Modelo: Oceanserver / Iver 2. Ecomapper

Latitude;Longitude;Time;Date;Number of Sats;GPS Speed (Kn);GPS True Heading;GPS Magnetic Variation;HDOP;C Magnetic Heading;C True Heading;Pitch Angle;Roll Angle;C Inside Temp (c);DFS Depth (m);DTB Height (m);Total Water Column (m);Batt Percent;Power Watts;Watt-Hours;Batt Volts;Batt Ampers;Batt State;Time to Empty;Current Step;Dist To Next (m);Next Speed (kn);Vehicle Speed (kn);Motor Speed CMD;Next Heading;Next Long;Next Lat;Next Depth (m);Depth Goal (m);Vehicle State;Error State;Fin Pitch R;Fin Pitch L;Pitch Goal;Fin Yaw T;Fin Yaw B;Yaw Goal;Fin Roll;DVL-Depth (m);DVL -Altitude (m);DVL -Water Column (m);Date m/d/y ;Time hh:mm:ss;Temp C;SpCond uS/cm;Sal ppt;Depth meters;pH;pH mV;Turbid+ NTU;Chl ug/L;Chl RFU;ODOsat %;ODO mg/L;

43.3906783333333;-8.29855333333333;03:19:09.66;10/19/2011;6;0.16;166.88;-3.28996025041067;1.6;80.1;76.8100397495893;-7.2;1.1;24.8;.05;999.99;999.99;92;20;432.2;15.7;-1.3;D;1363;0;4.498;0.5;0;128;136.4794;-8.298515;43.390649;0;0;1;N;48;48;15;250;250;136.4794;0;9999.99;999.99;999.99;10/19/11;08:15:22;17.45;2;0.00;0.169;7.49;-40.4;71.5;5.1;1.2;99.8;9.56;

43.390675;-8.29855166666667;03:19:10.66;10/19/2011;6;0.2;155.56;-3.28996025041067;1.6;79;75.7100397495893;-6.8;0;24.7;.03;999.99;999.99;92;20;432.1;15.7;-1.3;D;1381.8;0;4.139;0.5;0;128;134.2979;-8.298515;43.390649;0;0;1;N;48;48;15;250;250;134.2979;0;9999.99;999.99;999.99;10/19/11;08:15:23;17.45;2;0.00;0.170;7.49;-40.4;70.9;4.1;1.0;99.9;9.56;

Modelo: Hydroid Remus6000

software_revision, vehicle_temperature, heading_rate, vehicle_pressure, depth, depth_goal, obs_fluorometer, volts, current_amps, gfi_percent, pitch, pitch_goal, roll, thruster_rpm, thruster_rpm_goal, compass_heading, heading_goal, mission_time, days_since_1970, latitude, longitude, dr_latitude, dr_longitude, goal_latitude, goal_longitude, estimated_velocity, heading_offset, flags, thruster_cmd, pitch_cmd, rudder_cmd, pitch_fin_pos, rudder_fin_pos, objective_tot_number, objective_number, percent_cpu_usage, objective_index, leg_number, spare_slider_0, roll_rate, pitch_rate, faults, nav_mode, faults_2,

21, 32.6977768, -3.58798027, 706.304321, 0.159106255, 25, 4.62810993, 27.3867149, 2.37310672, 0, 1.51062012, -15, 4.58679199, 22, 188, 84.8693848, 49.0392151, 7:19:37.6, 14350, -4.82432146557, -12.3944680765, -4.82333327968, -12.3933327699, -4.82333333295, -12.3933333333, 0.179666668, 0, 11328, 9, 100, 100, 96, 100, 5, 2, 59, 13, 1, 0, -16.1424141, 3.30122161, 1660977152, 3, 16,

21, 32.6957397, 4.20346212, 706.364197, 0.159106255, 25, 4.63517284, 27.3275146, 3.26467466, 0, 1.07116699, -15, -2.16430664, 72, 188, 83.5070801, 49.1814575, 7:19:38.6, 14350, -4.82431869954, -12.3944705911, -4.82333285293, -12.3933291431, -4.82333333295, -12.3933333333, 0.588000059, 0, 11328, 19, 100, 100, 96, 101, 5, 2, 62, 13, 1, 0, 11.8340406, -2.49689865, 1660944384, 3, 18,

6.2. Ayuda en línea del programa AUVReport

MAIN MENU

The main main has the following options:

File: Load or generate a new report project.

- **New**: Generates a new project
- **Load**: Load a previous project that can be replayed / modified

Report: Report generation.

Tools: Tools for data analysis and quality control**.**

- DVL to Bathy
- QC
- Statistics
- Nav Correction

Database: Not active. For future use.

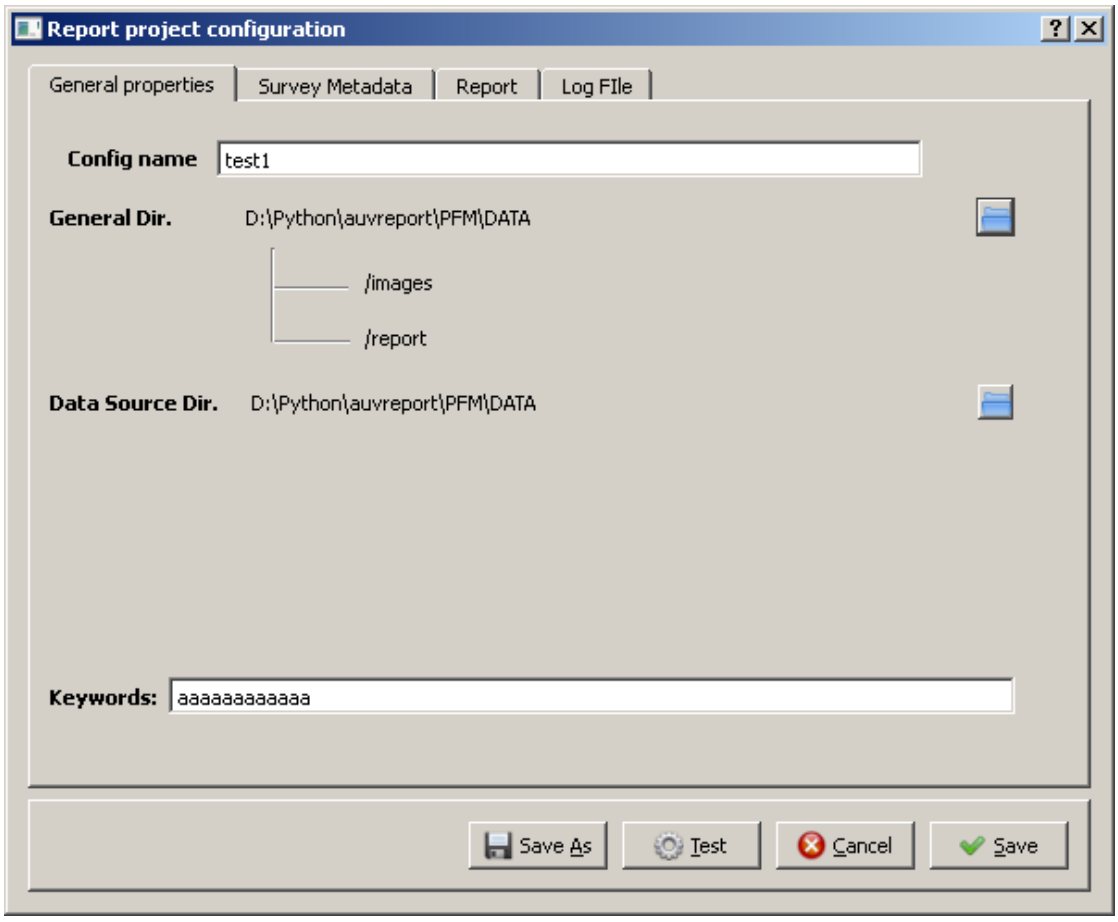
Help: Ayuda en línea

<< Main	**NEW / LOAD PROJECT**	

Generates a new project or loads a previous project to be modified.

The window is organized in tabs:

General Properties Tab:

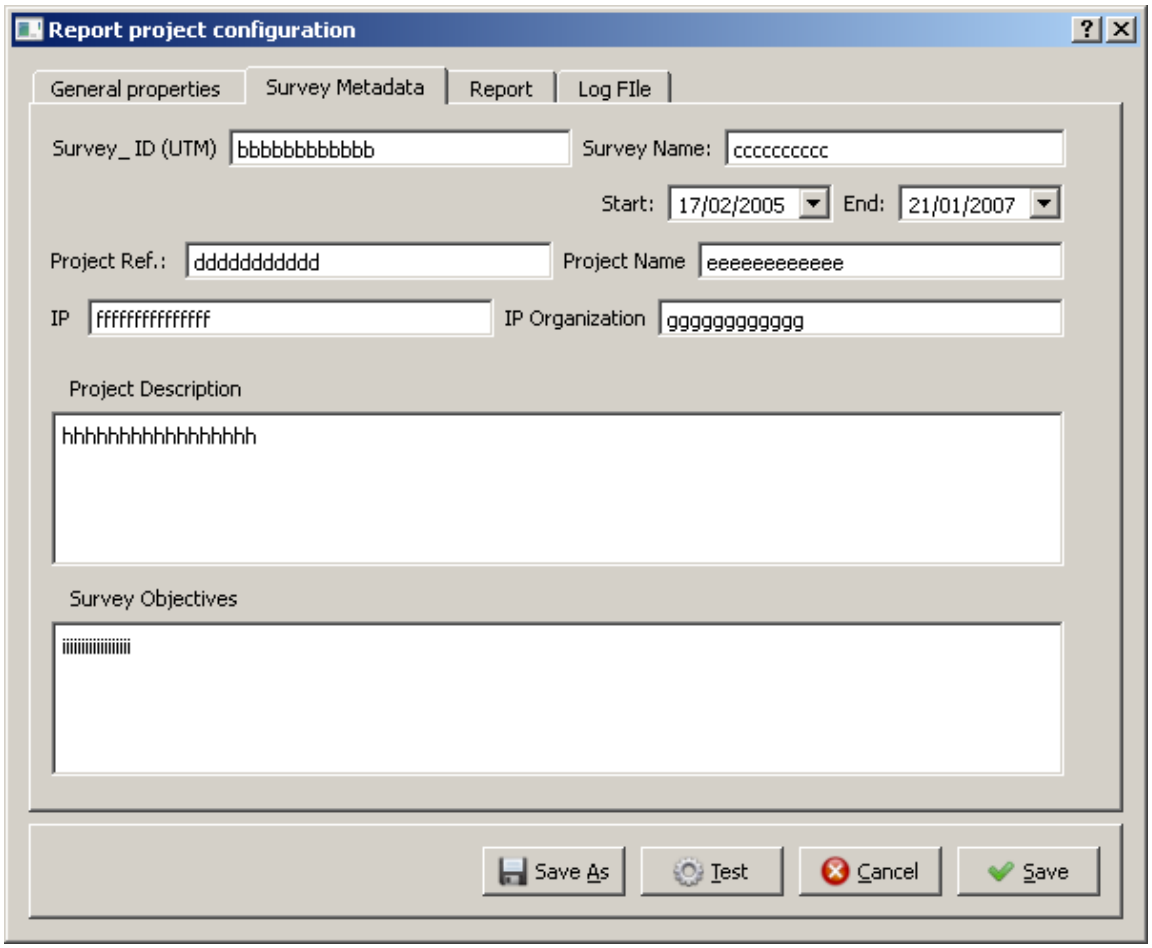


- **Config name:** Name of the XML File with all the config data.
- **General Dir.:** Main directory for the project files (configuration, images, report)
- **Data Source Dir**.: Data directoy
- **Keywords.**

Important notes:

Main (General) directory **MUST** exist before, however, child directories are created if don't exist.

Survey Metadata Tab:

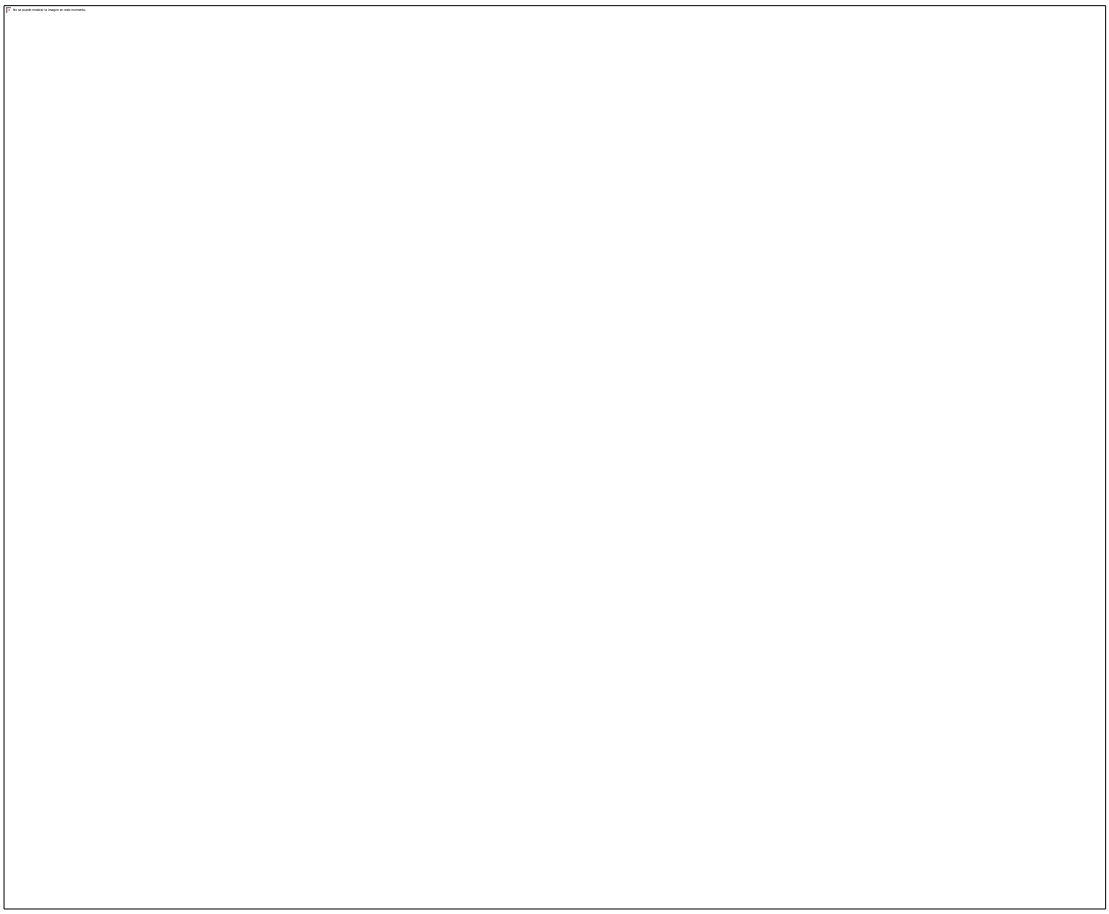


- **Survey_ID:** Internal code of the survey.
- **Survey Name:** General name / Acronim of the survey.
- **Star / End date:** Dates of the survey
- **Project reference / Name:** Reference ID and name of the project.
- **IP Name / Organization:** Name and organization of the Chief Scientist (IP).
- **Project Description:** Brief description of the project
- **Survey objectives:** Brief description of the AUV survey nside the Project objectives

Important notes:

Start Date must be earlier than *End Date*

Report Tab:



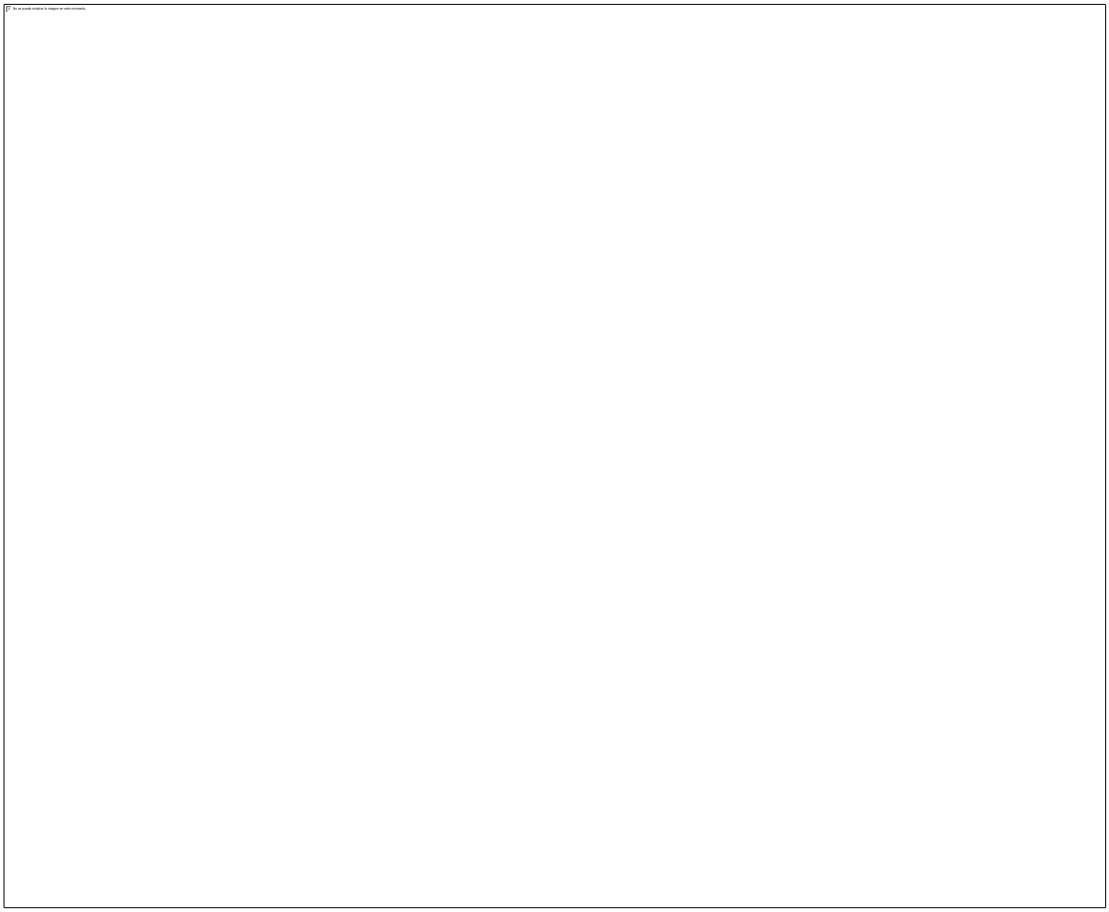
Generate Report: Activate if you want to generate a pdf report, it enables the following fields:

- **Author**: Report authoring.
- **Survey Area picture:** Activate if you want to include a general overview image of the area surveyed.
- **Statistics:** Activate if you want to include general statistics of the line surveyed.
- **Internal paremter plots:** Activate if you want to include plots selected on the *Log File Tab*.
- **Navigation:** Activate if you want to include navigation with depth.
- **Incidence file:** Activate if you want to include Incidence file (txt format only)
- **QC Graphs:** Not implemented (yet)
- **Calibration files:** Not implemented (yet).

Save Button: Save Configuration

Test Button: Test log files graphs.

Log File Tab:



Vehicle: Select the vehicle log file format (currently supports Iver and Remus 6000 format files)

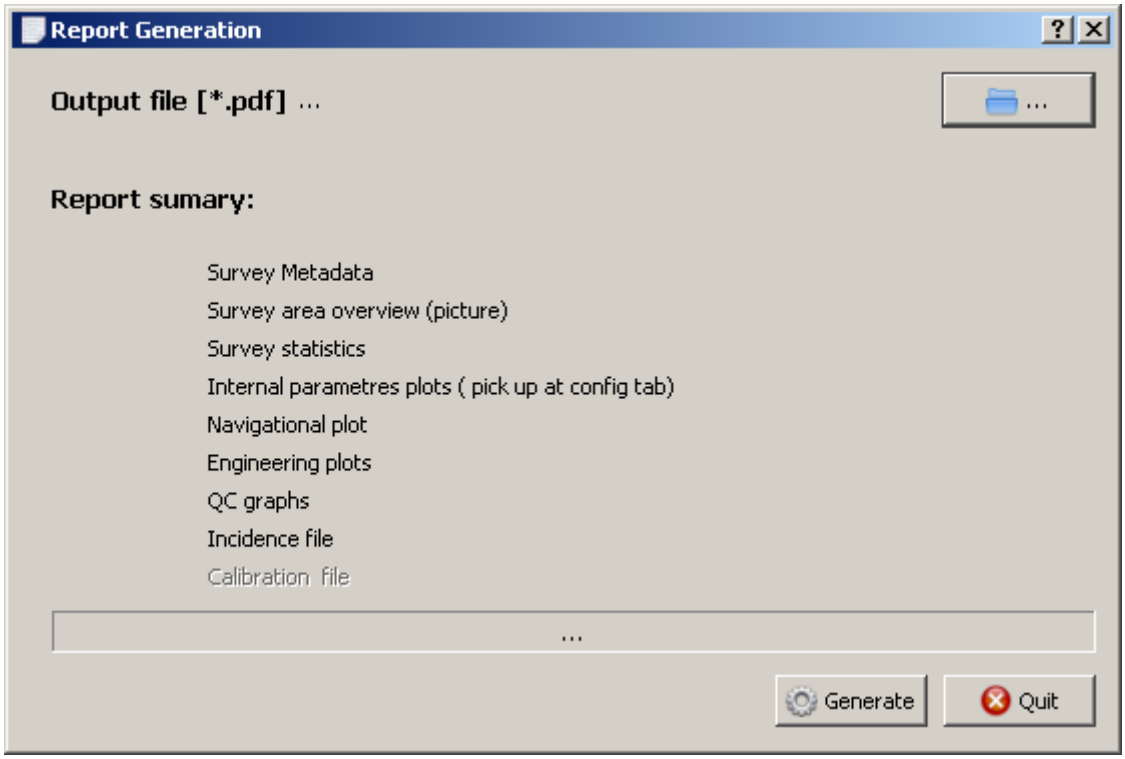
Read Log file Button: Select the log file to visualize

Actions:

Drag the variable name from the rightmost list to the X, Y fields on the table. Void Fields will be interpreted as a sequence number inside the file.

Tick the righmost checkbox to include selected plots on the report and visualize them with the TEST button

<< Main	**REPORT GENERATION**	



Launch report generation.

- **Output file [*.pdf]:** Name of the report PDF File.
- **Report Summary:** Data contained inside the report, if you wish to modify it quit and load the configuration again.

Generate Button: Generates the report, a DONE ! message appears when finished

Quit Button: Close the window

7. Referencias.

1 Gliderscope: http://imos.org.au/anfo_data.html
2 http://www.ivs3d.com/products/
3 http://www.goldensoftware.com
4 http://matplotlib.sourceforge.net/
5 http://code.enthought.com/projects/mayavi/
6 http://pyqwt.sourceforge.net
7 http://ccom.unh.edu/presentations/Schmidt_2009_CCOM_Seminar_AUV.pdf
8 http://epydoc.sourceforge.net/
9 http://www.python.org
10 http://docs.python.org/release/3.1.3/whatsnew/3.0.htm
11 http://www.reportlab.com/software/opensource/
12 http://eric-ide.python-projects.org/
13 http://sourceforge.net/projects/numpy/files/NumPy/
14 http://www.reportlab.com/software/installation/

BIBLIOGRAFÍA

S. Tosi. *Matplotlib for Python Developers.* Packt Publishing, 2006.
M. Summerfield. *Rapid GUI Programming with Python and Qt.* Prentice Hall.
T. Hall. Python 3 for Absolute Beginners. Apress 2009