PROGRAMMI PYTHON PER LA GENERAZIONE DELLE PAGINE WEB DEDICATE ALLA VISUALIZZAZIONE DEI DATI TECNICO-SCIENTIFICI DEL SEAGLIDER "AMERIGO"

P. ZUPPELLI, A. BUSSANI, R. GERIN, E. MAURI & P.-M. POULAIN

Approved by: ……

Dr. Paola Del Negro

INDICE:

1. Introduzione

Il glider, o aliante sottomarino, è uno strumento sviluppato allo scopo di monitorare vaste aree marine per periodi di osservazione che possono arrivare fino ad oltre 4 mesi. Si tratta di un veicolo autonomo, che però deve essere continuamente monitorato ed i cui parametri devono essere frequentemente modificati al fine di garantire contemporaneamente sia la sorveglianza dello strumento, che la buona riuscita della missione.

A questo scopo, si sono sviluppati alcuni programmi che risiedono sul server **oceano**, i quali elaborano i dati che il glider invia a “terra” e salvano i loro prodotti sul server **nettuno**. In questo modo si riuscirà ad ottenere una panoramica sempre aggiornata, dello stato dello strumento e uno storico dell’andamento delle sue posizioni anche via web. In particolare, questi programmi, si inseriscono nella catena di elaborazione generale dei dati del glider tra il server **oceano** ed il server **nettuno** (A. Bussani and R. Gerin, 2013).

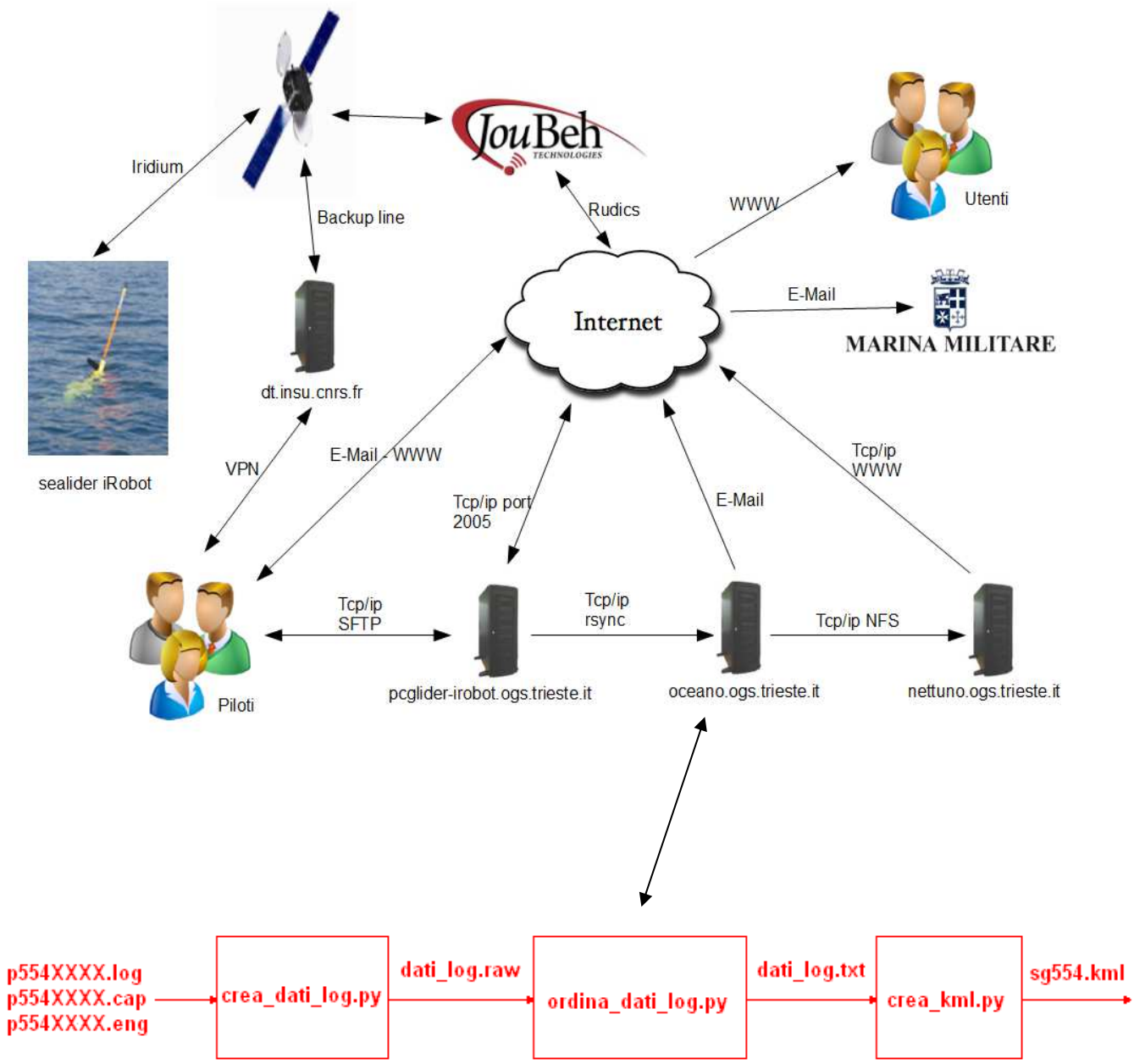

Fig.1: Diagramma di flusso della gestione dei dati integrato in figura pag. 19 di Bussani and Gerin (2013).

2. Parametri per la navigazione.

Il programma che si occupa dell'estrazione delle informazioni utili alla navigazione e al monitoraggio dello stato dello strumento è **crea_dati_log.py** il quale risiede nel server **oceano** nella cartella: **/storage/sire/dati/glider/seaglider/amerigo/actual**. Questo programma cerca all'interno dei file dati (i quali sono **p554XXXX.log**, **p554XXXX.cap** e **p554XXXX.eng** dove le XXXX rappresentano in numero di dive) determinati campi che contengono informazioni come numero di missione e numero di dive, punti GPS acquisiti all'inizio e alla fine del dive, stato delle batterie, pressione e umidità interna dello strumento, prossimo way-point attivo con le relative coordinate, lo stato della memoria interna, sul comportamento delle varie componenti sensoriali, eventuali errori nella connessione, nella scrittura dei dati e sul comportamento delle varie componenti sensoriali.

I dati estratti vengono salvati in un file chiamato **dati_log.raw,** nella cartella condivisa tra il server **oceano** e il server **nettuno (/storage/sire/work/web/sire/glider)**, allo scopo di avere uno storico dei dati acquisiti e permettere le ulteriori elaborazioni per poi visualizzare gli stessi dati ondine, mediante delle pagine web.

I dati così estratti dai file vengono salvati con la creazione di una stringa nel seguente modo:

1 25 220312 125723 47.7258333333 -122.390316667 220312 130359 47.7254333333 -122.390183333 220312 133855 47.7276666667 -122.392866667 10.3 4.087 24.2 4.619 18.97 9.01359 2097086464 2089975808 5 47.725 -122.391666667 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 160.589203 22-Mar-201213:05:06

stringa[1] = numero della missione
stringa[2] = numero del dive
stringa[2] = data del primo punto gps

stringa[3] = ora del primo punto gps

stringa[4] = longitudine del primo punto gps

stringa[5] = latitudine del primo punto gps

stringa[6] = data del secondo punto gps

stringa[7] = ora del secondo punto gps

stringa[8] = longitudine del secondo punto gps

stringa[9] = latitudine del secondo punto gps

stringa[10] = data dell'ultimo punto gps acquisito

stringa[11] = ora dell'ultimo punto gps acquisito

stringa[12] = longitudine dell'ultimo punto gps acquisito

stringa[13] = latitudine dell'ultimo punto gps acquisito

stringa[14] = stato della carica della batteria da 10V

stringa[15] = ampere della batteria da 10V

stringa[16] = stato della carica della batteria da 24V

stringa[17] = ampere della batteria da 24V

stringa[18] = umidità interna dello strumento

stringa[19] = pressione interna dello strumento

stringa[20] = capacità totale della memoria

stringa[21] = spazio disponibile sulla memoria

stringa[22] = nome del target

stringa[23] = longitudine del target

stringa[24] = latitudine del target

stringa[25] = errore buffer overflow

stringa[26] = errore

stringa[27] = numero errori di cf8 in apertura

stringa[28] = numero errori di cf8 in scrittura

stringa[29] = numero errori di cf8 in chiusura

stringa[30] = numero tentativi di cf8 in apertura

stringa[31] = numero tentativi di cf8 in scrittura

stringa[32] = numero tentativi di cf8 in chiusura

stringa[33] = numero errori pitch

stringa[34] = numero errori roll
stringa[35] = numero errori VBD
stringa[36] = numero tentativi pitch
stringa[37] = numero tentativi roll
stringa[38] = numero tentativi VBD
stringa[39] = numero di timeout del $GPSRMC
stringa[40] = profondità a cui è stato trovato il fondale (se 9999, non è stato trovato il file **.cap**, se 0 non è stato trovato il fondale).
stringa[41] = data e ora di inizio del dive

Tab.1: Descrizione dei singoli campi che costituiscono la stringa dati

E' stata anche prevista la possibilità che, alcuni file necessari alla creazione della stringa oppure dei singoli dati siano assenti (per errori di trasmissione dei file): in questo caso il programma inserisce degli 0.

3. Trattamento dei dati estratti

Può accadere che il glider non trasmetta sempre tutti i file necessari per il monitoraggio e il pilotaggio dello strumento in modo corretto, quindi si è previsto che possano arrivare informazioni non ordinate in modo cronologico. Il programma che si occupa di risolvere questo inconveniente ed eventualmente eliminare dati "doppi" presenti nel file di dati **dati_log.raw** è l'**ordina_dati_log.py** ed anche questo programma, come il **crea_dati_log.py**, si trova nella cartella **/storage/sire/dati/glider/seaglider/amerigo/actual** sul server **oceano**.

Ordina_dati_log.py è un programma diviso sostanzialmente in due parti:

La prima parte legge il file dati_log.raw, lo ripulisce da ogni carattere indesiderato (quali parentesi virgole e apici), lo divide in una matrice di dati suddivisa in righe, che corrispondono al dato del singolo dive, e colonne per i vari campi della stringa dati. A

questo punto il programma ordina la matrice creata in base al valore della prima colonna di ogni riga della matrice e cioè per il numero di missione; in seguito viene eseguito un secondo riordino per numero di dive (seconda colonna), in modo da avere i dati potenzialmente simili (stessa missione e stesso dive) vicini tra loro e salva il tutto in un file temporaneo chiamato **dati_log_tmp.txt** nella cartella **/storage/sire/work/web/sire/glider** del server oceano.

A causa di alcuni ritardi o errori nella trasmissione, non sempre tutti i file dati arrivano a terra. Per questo motivo, la seconda parte del programma, legge il file temporaneo appena creato, due righe consecutive alla volta, divide ogni riga in campi e confronta determinati valori (numero di missione, numero di dive e, se è stato rilevato il fondale). In questo modo, si esclude la possibilità di avere dei dati parziali (dovuti ad elaborazioni effettuate in ritardo) o delle loro ripetizioni nel file finale.

Infine il programma genera una stringa di testo contenente la latitudine e longitudine dell'ultimo punto rilevato dal glider e il prossimo way-point.

Questa stringa, verrà spedita via email ad una lista di indirizzi precedentemente impostati in modo da avere un'ulteriore controllo sullo stato dello strumento. (vedi Fig.3).

Ultima posizione del 17-04-2013 11:44:54 UTC : lat/lon 45.7105333–33 13.7625666667 - waypoint attivo: lat/lon 41.5691666667 17.2291666667

Fig.3 Esempio di stringa inviata via email.

4. Creazione del file .kml per la visualizzazione della rotta su google earth

Per rendere più comodo il pilotaggio del glider, si è deciso di sviluppare un'ulteriore programma per automatizzare la creazione di un file per permettere la visualizzazione dei punti acquisiti e della rotta seguita dal glider tramite google earth.

Il programma che si occupa di questo aspetto è il **crea_kml.py.** Esso genera un file con estensione **.kml** (estensione dei file per la visualizzazione su google earth),

contenente le definizioni gli stili e le icone (che si trovano sul server **nettuno**, nella cartella **/sire/google_earth/glider/)** che verranno utilizzate per la visualizzazione della rotta seguita dal glider e contenente i vari punti intermedi di affioramento.

```
<Placemark>
    <styleUrl>#SubmergeStyler</styleUrl>
    <gx:balloonVisibility>1</gx:balloonVisibility>
    <description><![CDATA[sg554<br>mission sumber:6<br>dive number:2<br>first GPS<br>2013-04-19 13:13:29 <br>45.7105166667<br>13.7625833333]]></description>
    <TimeStamp>
        <when>2013-04-19T13:13:29Z</when>
    </TimeStamp>
    <Point>
        <coordinates>13.7625833333,45.7105166667,0</coordinates>
    </Point>
</Placemark>
```

Fig.4 Porzione di codice che definisce i punti di affioramento.

Successivamente, il programma procede con una prima lettura del file dati_log.txt, estrae le coordinate dei punti in superficie acquisiti e li salva su un file di nome **sg554.kml** in nettuno, nella cartella **/storage/sire/work/web/sire/glider/**. In fine una seconda lettura del file **dati_log.txt** permette di creare le traiettorie seguite dal glider sott'acqua e in superficie, assegnando due colori diversi.

```
<Placemark>
    <styleUrl>#lineStyle</styleUrl>
    <TimeSpan>
        <begin>2013-04-19T12:57:37Z</begin>
        <end>2013-04-19T13:13:29Z</end>
    </TimeSpan>
```

```
        <LineString>
                <tessellate>1</tessellate>
                <coordinates>
                        13.7625333333,45.7105,0
                        13.7625833333,45.7105166667,0
                </coordinates>
        </LineString>
</Placemark>
```

Fig.5 Porzione di codice che definisce le traiettorie effettuate dal glider.

Allo scopo di rendere più semplice la visualizzazione delle traiettorie, si è pensato di utilizzare un'animazione particolare del Google Earth, la struttura timespan, che permette di visualizzare le rotte separate in periodi di tempo (in modo da poter visualizzare solamente una particolare lasso di tempo di missione o tutta la rotta nel suo insieme). Per questo motivo, nella struttura del file **.kml**, vengono specificate le date e le ore dei punti gps acquisiti.

Fig 6. Esempio di visualizzazione delle rotte del glider tramite Google earth.

5. Riferimenti/Bibliografia/Relazioni

http://nettuno.ogs.trieste.it/sire/glider/Seaglider%20User%20Guide%20-%20Rev.%20C.pdf
http://nettuno.ogs.trieste.it/sire/glider/appunti_seattle_elena.pdf
http://nettuno.ogs.trieste.it/sire/glider/appunti_seattle_riccardo.pdf
https://developers.google.com/kml/documentation/topicsinkml
http:// http://www.python.org/doc/

Bussani A. and Gerin R. (2013) Configurazione e gestione del flusso dati del seaglider iRobot OGS2013/45 Sez. OCE 22 MAOS 33

Appendix A

```
#!/usr/bin/python
#
#------------------------------ Include files --------------------------------
#
import sys
import os, array, sys, glob
import math


errori = "$ERRORS"
batt_24V = "$24V_AH"
batt_10V = "$10V_AH"
last_pos = "$GPS"
mission_num = "$MISSION"
dive_num = "$DIVE"
gps1 = "$GPS1"
gps2 = "$GPS2"
tgt_name = "$TGT_NAME"
tgt_latlon = "$TGT_LATLONG"
humid = "$HUMID"
inter_p = "$INTERNAL_PRESSURE"
cfsize = "$CFSIZE"
file_dati = "/storage/sire/work/web/sire/glider/dati_log.raw"
bott_dec = "obstacle/bottom"
start_dive = "start"

#------------------------------------------------------------------------------
#------------------------------- Subroutine -----------------------------------
#------------------------------------------------------------------------------

#------------------------------ cerca_campo -----------------------------------

def cerca_campo(file, nome_campo): #funzione che cerca il campo passato dal programma
  f = open(file, 'r')
  a = 0
```

```
    while 1:
            line = f.readline()
            if not line: break
            riga = line.split(",")
            if riga[0] == nome_campo:
                    if nome_campo == gps1:   # dei vari gps ci interessano piu campi dalla stessa riga
                            a = 1
                            return riga[1:5]
                    if nome_campo == gps2:
                            a = 1
                            return riga[1:5]
                    if nome_campo == last_pos:
                            a = 1
                            return riga[1:5]
                    else:
                            a = 1
                            return riga[1:]
    if a == 0:
            return -1
    f.close()

def cerca_obstacle(file, nome_campo):
    f = open(file, 'r')
    a=0
    while 1:
            line = f.readline() #legge tutte le linee e cerca il campo che ci interessa
            if not line: break
            trova = line.find(nome_campo)
            if trova > 0:
                    riga = line.split(" ")
                    a = riga[-2]          #cerco il valore della profondita a cui ha trovato il fondo
    f.close()
    return a

def cerca_start_dive(file, nome_campo):
    f = open(file, 'r')
    a=0
    while 1:
            line = f.readline() #legge tutte le linee e cerca il campo che ci interessa
            if not line: break
```

```
                trova = line.find(nome_campo)
                if trova > 0:
                        riga = line.split(": ")
                        start = riga[-1]
                        start = start.replace("\n", '')
                        start = start.split(" ")
                        a = start[1] + "-" + start[0] + "-" + start[2] + "_" + start[3] + ":" + start[4] + ":" + start[5]
#cerco il valore della profondita a cui ha trovato il fondo
          f.close()
          return a

        #------------------------------ crea_percorso ---------------------------------

        def crea_percorso(nome_file): #crea il nome del file .txt dal nome del file .com
          new_file = nome_file.split("/")
          nome = new_file[-1]
          new_percorso = nome_file.replace(nome, '')
          return new_percorso

        #------------------------------ crea_linea ---------------------------------

        def crea_linea(linea_arr):
          linea_str = str(linea_arr)
          linea_str = linea_str.replace('[', '')
          linea_str = linea_str.replace("'", '')
          linea_str = linea_str.replace(',', '')
          linea_str = linea_str.replace('n', '')
          linea_str = linea_str.replace(unichr(92), '')
          linea_str = linea_str.replace(']', '')
          return linea_str

        #------------------------------ crea_gps ---------------------------------

        def crea_gps(gps_array):
          lon_100 = float(gps_array[2])/100
          lat_100 = float(gps_array[3])/100
          lon_int = int(lon_100)
          lat_int = int(lat_100)
          lon_sec = (lon_100 - lon_int)*100/60
          lat_sec = (lat_100 - lat_int)*100/60
```

```
  lon = lon_int + lon_sec
  lat = lat_int + lat_sec
  gps_array[2] = str(lon)
  gps_array[3] =str(lat)
  return gps_array

#------------------------------- crea_gps_tgt ----------------------------------

def crea_gps_tgt(gps_array):
  lon_100 = float(gps_array[1])/100
  lat_100 = float(gps_array[0])/100
  lon_int = int(lon_100)
  lat_int = int(lat_100)
  lon_sec = (lon_100 - lon_int)*100/60
  lat_sec = (lat_100 - lat_int)*100/60
  lon = lon_int + lon_sec
  lat = lat_int + lat_sec
  gps_array[1] = str(lon)
  gps_array[0] =str(lat)
  return gps_array



#------------------------------- crea file log ----------------------------------

def crea_log(testo_log):
  testlog = open("testlog_creadatilog.txt", "a")
  testlog.write(testo_log)
  testlog.close()
  return 0

#---------------------------------- Main ------------------------------------

#--------------------------- controllo file ingresso ---------------------------

file_in = len(sys.argv)                    #valuta quantita dati d'ingresso
if file_in==2:
  file_log = sys.argv[1]         #legge il campo relativo al file di log
elif file_in>3:
  print 'crea_dati_log: ci sono troppi file in ingresso e uscita'
```

```
    sys.exit(1)
else:
    print 'crea_dati_log: manca il file di ingresso e/o di uscita'
    sys.exit(1)

esisti_log = str(os.path.exists(file_log))
crea_log(file_log + " " + esisti_log +" ")

name_log = file_log.split(".")
file_cap = name_log[0] + ".cap"
crea_log(file_cap + " ")

esisti_cap = os.path.exists(file_cap)  #controllo esistenza file d'ingresso eng
crea_log(str(esisti_cap) + " ")
if os.path.exists(file_log):
    #-------------------- controllo campi log e creazione file --------------------

    file_dati_log = file_dati
    campo_mission_num = cerca_campo(file_log, mission_num)
    campo_dive_num = cerca_campo(file_log, dive_num)
    campo_gps1 = cerca_campo(file_log, gps1)
    campo_gps2 = cerca_campo(file_log, gps2)
    campo_last_pos = cerca_campo(file_log, last_pos)
    campo_batt_10V = cerca_campo(file_log, batt_10V)
    campo_batt_24V = cerca_campo(file_log, batt_24V)
    campo_humid = cerca_campo(file_log, humid)
    campo_inter_p = cerca_campo(file_log, inter_p)
    campo_cfsize = cerca_campo(file_log, cfsize)
    campo_tgt_name = cerca_campo(file_log, tgt_name)
    campo_tgt_latlon = cerca_campo(file_log, tgt_latlon)
    campo_errori = cerca_campo(file_log, errori)

    new = open(file_dati_log, "a")
    if campo_mission_num == -1:
            new.write("0" + " ")
    else:
            new.write(crea_linea(cerca_campo(file_log, mission_num)) + " ")
    if campo_dive_num == -1:
            new.write("0" + " ")
    else:
```

```
            new.write(crea_linea(cerca_campo(file_log, dive_num)) + " ")
    if campo_gps1 == -1:
            new.write("0" + " " + "0" + " " + "0" + " " + "0" + " ")
    else:
            new.write(crea_linea(crea_gps(cerca_campo(file_log, gps1))) + " ")
    if campo_gps2  == -1:
            new.write("0" + " " + "0" + " " + "0" + " " + "0" + " ")
    else:
            new.write(crea_linea(crea_gps(cerca_campo(file_log, gps2))) + " ")
    if campo_last_pos == -1:
            new.write("0" + " " + "0" + " " + "0" + " " + "0" + " ")
    else:
            new.write(crea_linea(crea_gps(cerca_campo(file_log, last_pos))) + " ")
    if campo_batt_10V == -1:
            new.write("0" + " " + "0" + " ")
    else:
            new.write(crea_linea(cerca_campo(file_log, batt_10V)) + " ")
    if campo_batt_24V == -1:
            new.write("0" + " " + "0" + " ")
    else:
            new.write(crea_linea(cerca_campo(file_log, batt_24V)) + " ")
    if campo_humid == -1:
            new.write("0" + " ")
    else:
            new.write(crea_linea(cerca_campo(file_log, humid)) + " ")
    if campo_inter_p == -1:
            new.write("0" + " ")
    else:
            new.write(crea_linea(cerca_campo(file_log, inter_p)) + " ")
    if campo_cfsize == -1:
            new.write("0" + " " + "0" + " ")
    else:
            new.write(crea_linea(cerca_campo(file_log, cfsize)) + " ")
    if campo_tgt_name == -1:
            new.write("0" + " ")
    else:
            new.write(crea_linea(cerca_campo(file_log, tgt_name)) + " ")

    if campo_tgt_latlon == -1:
            new.write("0" + " " + "0" + " ")
```

```
        else:
                new.write(crea_linea(crea_gps_tgt(cerca_campo(file_log, tgt_latlon))) + " ")
        if campo_errori == -1:
                new.write("0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " " + "0" + " ")
        else:
                new.write(crea_linea(cerca_campo(file_log, errori)) + " ")

        name = file_log.split(".")
        file_cap = name[0] + ".cap"
        file_eng = name[0] + ".eng"

        crea_log(file_eng + " ")
        esisti_eng = os.path.exists(file_eng)           #controllo esistenza file d'ingresso eng
        crea_log(str(esisti_eng) + " ")
        if os.path.exists(file_cap):
                new.write(str(cerca_obstacle(file_cap, bott_dec)) + " ")
        else:
                new.write("9999" + " ")
        if os.path.exists(file_eng):
                new.write(str(cerca_start_dive(file_eng, start_dive)) + "\r\n")
        else:
                new.write("no_data_dive" + "\r\n")

        new.close()
        crea_log("\r\n")
      else:
        print "crea_dati_log: " + file_log + " : input file not found"
        sys.exit(1)
```

Appendix B

```
#!/usr/bin/python
#
#------------------------------ Include files ---------------------------------
#
import os, array, sys
import sets

#-------------------------------------------------------------------------------
#--------------------------------- Subroutine ----------------------------------
#-------------------------------------------------------------------------------

#---------------------- Ordina file e elimina doppioni -----------------------

def sort(filein, fileoutraw, fileout):
  f = open(filein, 'r')
  lista = []
  for line in f:
        line = line.strip()
        nome = tuple(p.strip() for p in line.split(" "))
        lista.append(nome)
  new = list(set(lista))
  new.sort(key=lambda nome: int(nome[1]))
  new.sort(key=lambda nome: int(nome[0]))
  file_out = open(fileoutraw, 'w')
  dati = str(new)
  dati = dati.replace('),', "\r\n")
  dati = dati.replace('[', "")
  dati = dati.replace(']', "")
  dati = dati.replace(',', "")
  dati = dati.replace("'", "")
  dati = dati.replace(' (', "")
  dati = dati.replace('(', "")
  dati = dati.replace(')', "")
  file_out.write(str(dati))
  file_out.close()
```

```
        f.close()
        file_raw = open(fileoutraw, 'r')
        number_lines = len(file_raw.readlines())
        file_raw.close()
        i=0
        crea_log("inizio " + "\r\n", "testlog_ciclo.txt")
        file_raw = open(fileoutraw, 'r')
        f = open(fileout, 'w')
        linea1 = file_raw.readline()
        campo_l1 = linea1.split(" ")
        campo0_l1 = campo_l1[0]
        campo1_l1 = campo_l1[1]
        campo40_l1 = campo_l1[40]
        crea_log("numero linee totali nel file " + str(number_lines) + "\r\n", "testlog_ciclo.txt")
        crea_log("dati riga " + str(i) + " campi: " + str(campo0_l1) + ";" + str(campo1_l1) + ";" + str(campo40_l1) +
";" + "\r\n", "testlog_ciclo.txt")
        while i<number_lines:
                i=i+1
                linea2 = file_raw.readline()
                if not linea2:
                        crea_log("linea numero " + str(i) + " inesistente, ciclo interrotto" + "\r\n", "testlog_ciclo.txt")
                        f.write(linea1)
                        f.close()
                        file_raw.close()
                        break
                crea_log("leggo la riga numero: " + str(i) + "\r\n", "testlog_ciclo.txt")
                campo_l2 = linea2.split(" ")
                campo0_l2 = campo_l2[0]
                campo1_l2 = campo_l2[1]
                campo40_l2 = campo_l2[40]
                crea_log("dati riga " + str(i) + " campi: " + str(campo0_l2) + ";" + str(campo1_l2) + ";" +
str(campo40_l2) + ";" + "\r\n", "testlog_ciclo.txt")
                crea_log("confronto linea " + str(i-1) + " con la linea " + str(i) + "\r\n", "testlog_ciclo.txt")
                if campo0_l1 == campo0_l2:
                        crea_log("numero missione uguale tra le due linee " + "\r\n", "testlog_ciclo.txt")
                        if campo1_l1 == campo1_l2:
                                crea_log("dive uguale tra le due linee " + "\r\n", "testlog_ciclo.txt")
                                if campo40_l1[:4] != '9999':
                                        crea_log("campo diverso da 9999: " + str(campo40_l1[:4]) + " scrivo " +
str(i-1) + "\r\n", "testlog_ciclo.txt")
                                        f.write(linea1)
```

```
                                        linea1 = file_raw.readline()
                                        if not linea1: break
                                        campo_l1 = linea1.split(" ")
                                        campo0_l1 = campo_l1[0]
                                        campo1_l1 = campo_l1[1]
                                        campo40_l1 = campo_l1[40]
                                        i=i+1
                                else:
                                        f.write(linea2)
                                        crea_log("campo uguale a: " + str(campo40_l1[:4]) + "  scrivo " + str(i) +
"\r\n", "testlog_ciclo.txt")
                                        linea1 = file_raw.readline()
                                        if not linea1: break
                                        campo_l1 = linea1.split(" ")
                                        campo0_l1 = campo_l1[0]
                                        campo1_l1 = campo_l1[1]
                                        campo40_l1 = campo_l1[40]
                                        i=i+1
                        else:
                                f.write(linea1)
                                crea_log("numero dive diverso, scrivo la linea precedente e passo alla
successiva " + "\r\n", "testlog_ciclo.txt")
                                linea1 = linea2
                                campo0_l1 = campo0_l2
                                campo1_l1 = campo1_l2
                                campo40_l1 = campo40_l2
                else:
                        f.write(linea1)
                        crea_log("numero missione diverso, scrivo la linea precedente e passo alla successiva "
+ "\r\n", "testlog_ciclo.txt")
                        linea1 = linea2
                        campo0_l1 = campo0_l2
                        campo1_l1 = campo1_l2
                        campo40_l1 = campo40_l2
        f.close()
        file_raw.close()
        #print fileoutraw
        #if os.path.exists(fileoutraw) != 0:
        #       os.remove(fileoutraw)

      #------------------------------ crea file log ----------------------------------
```

```
def crea_log(testo_log, logfile):
  testlog = open(logfile, "a")
  testlog.write(testo_log)
  testlog.close()
  return 0

#---------------------------------- main --------------------------------


file_in = len(sys.argv)                    #valuta quantita dati d'ingresso
if file_in==2:
  file_raw = sys.argv[1]       #legge il campo relativo al file di log


esisti_raw = str(os.path.exists(file_raw))
crea_log(file_raw + " " + esisti_raw + " ", "testlog_ordinadatilog.txt")

if os.path.exists(file_raw):               #controllo esistenza file d'ingresso log
  name_raw = file_raw.split(".")
  file_out_tmp = name_raw[0] + "_tmp.txt"
  crea_log(file_out_tmp + " ", "testlog_ordinadatilog.txt")
  file_out = "/storage/sire/work/web/sire/glider/dati_log.txt"
  control = "/storage/sire/work/glider/web/amerigo/actual/controllo_email.txt"
  sort(file_raw, file_out_tmp, file_out)
  f = open(file_out, "r")
  lastline = f.readlines()[-1 : ]
  campo = str(lastline).split(" ")
  f.close()
  gps_last_time = campo[11]
  gps_last_date = campo[10]
  gps_last_lat = campo[12]
  gps_last_lon = campo[13]
  data_ora = gps_last_date[0:2] + "-" + gps_last_date[2:4] + "-" + "20" + gps_last_date[4:6] + " " + gps_last_time[0:2] + ":" + gps_last_time[2:4] + ":" + gps_last_time[4:6]
  tgt_lat = campo[23]
  tgt_lon = campo[24]
  if os.path.exists(control):
        file_control = open(control, "r")
        line_control = file_control.readlines()[-1 : ]
```

```
                file_control.close()
                line_control = str(line_control)
                line_control = line_control.replace("['", "")
                line_control = line_control.replace("']", "")
                if line_control == data_ora:
                        stringa = "echo " + "Ultima posizione del " + data_ora + " UTC : " + "lat/lon " + gps_last_lat + "  " + gps_last_lon + " - " + "waypoint attivo: "  + "lat/lon " + tgt_lat + " " + tgt_lon
                else:
                        file_control = open(control, "w")
                        file_control.write(data_ora)
                        file_control.close()
                        stringa = "echo " + "Ultima posizione del " + data_ora + " UTC : " + "lat/lon " + gps_last_lat + "  " + gps_last_lon + " - " + "waypoint attivo: "  + "lat/lon " + tgt_lat + " " + tgt_lon
                        os.system(stringa)
        else:
                file_control = open(control, "w")
                file_control.write(data_ora)
                file_control.close()
                stringa = "echo " + "Ultima posizione del " + data_ora + " UTC : " + "lat/lon " + gps_last_lat + "  " + gps_last_lon + " - " + "waypoint attivo: "  + "lat/lon " + tgt_lat + " " + tgt_lon
                os.system(stringa)
        crea_log("\r\n", "testlog_ordinadatilog.txt")
      else:
        print "ordina_dati_log" + file_raw + " : input file not found"
        sys.exit(1)
```

Appendix C

```
#!/usr/bin/python
#
#------------------------------ Include files --------------------------------
#
import sys
import os, array, sys, glob, re


#-----------------------------------------------------------------------------
#-------------------------------- Subroutine ---------------------------------
#-----------------------------------------------------------------------------
#-----------------------------------struttura kml-----------------------------------
def struttura(nome, intestazione, m_num, d_num, lat, lon, date, style):
  f = open(file_kml, 'a')
  date_desc = date.replace("T", " ")
  date_desc = date_desc.replace("Z", " ")
  f.write("\t" + '<Placemark>' + "\r\n")
  f.write("\t" + "\t" + '<styleUrl>'+ style + '</styleUrl>' + "\r\n")#
  f.write("\t" + "\t" + '<gx:balloonVisibility>1</gx:balloonVisibility>' + "\r\n")#
  f.write("\t" + "\t" + "<description><![CDATA[" + nome + "<br>" + "mission number:" + m_num + "<br>" + "dive number:" + d_num + "<br>" + intestazione + "<br>" + date_desc + "<br>" + lat + "<br>" + lon + "]]></description>" + "\r\n")
  f.write("\t" + "\t" + '<TimeStamp>' + "\r\n")
  f.write("\t" + "\t" + "\t" + "<when>" + date + "</when>" + "\r\n")
  f.write("\t" + "\t" + "</TimeStamp>" + "\r\n")
  f.write("\t" + "\t" + "<Point>" + "\r\n")
  f.write("\t" + "\t" + "\t" + "<coordinates>" + lat + "," + lon + "," + "0" + "</coordinates>" + "\r\n")
  f.write("\t" + "\t" + '</Point>' + "\r\n")
  f.write("\t" + '</Placemark>' + "\r\n")
  f.close()


def struttura_next(nome, intestazione, lat, lon):
  f = open(file_kml, 'a')
  f.write("\t" + '<Placemark>' + "\r\n")
  f.write("\t" + "\t" + '<styleUrl>#NextStyler</styleUrl>' + "\r\n")#
  f.write("\t" + "\t" + '<gx:balloonVisibility>1</gx:balloonVisibility>' + "\r\n")#
  f.write("\t" + "\t" + "<description><![CDATA[" + nome + "<br>" + intestazione + "<br>" + lat + "<br>" + lon + "]]></description>" + "\r\n")
  f.write("\t" + "\t" + "<Point>" + "\r\n")
```

```
    f.write("\t" + "\t" + "\t" + "<coordinates>" + lat + "," + lon + "," + "0" + "</coordinates>" + "\r\n")
    f.write("\t" + "\t" + '</Point>' + "\r\n")
    f.write("\t" + '</Placemark>' + "\r\n")
    f.close()


def linestring (stile, timebegin, timeend, coordinate1, coordinate2):
    f = open(file_kml, 'a')
    f.write("\t" + '<Placemark>' + "\r\n")
    f.write("\t" + '<styleUrl>#' + stile + '</styleUrl>' + "\r\n")
    f.write("\t" + "\t" + '<TimeSpan>' + "\r\n")
    f.write("\t" + "\t" + "\t" + "<begin>" + timebegin + "</begin>" + "\r\n")
    f.write("\t" + "\t" + "\t" + "<end>" + timeend + "</end>" + "\r\n")
    f.write("\t" + "\t" + '</TimeSpan>' + "\r\n")
    f.write("\t" + '<LineString>' + "\r\n")
    f.write("\t" + "\t" + '<tessellate>1</tessellate>' + "\r\n")
    f.write("\t" + "\t" + '<coordinates>' + "\r\n" + "\t" + "\t")
    f.write(coordinate1)
    f.write("\r\n")
    f.write(coordinate2)
    f.write("\r\n")
    f.write("\t" + "\t" + '</coordinates>' + "\r\n")
    f.write("\t" + '</LineString>' + "\r\n")
    f.write("\t" + '</Placemark>' + "\r\n")
    f.close()


#-------------------------------------------------------------------------------
def crea_log_kml(file_kml, filelog):#, head_desc):
    esisti_kml = os.path.exists(file_kml)
    if esisti_kml == 'True':
            os.remove(file_kml)
    f = open(file_kml, 'a')
    #---------------------- creo intestazione ----------------------------
    f.write('<?xml version="1.0" encoding="UTF-8" ?>' + "\r\n" + '<kml
xmlns="http://earth.google.com/kml/2.1">' + "\r\n" + '<Document xmlns:xlink="http://www.w3/org/1999/xlink">' +
"\r\n")
    f.write("\t" + '<Style id="lineStyle">' + "\r\n")
    f.write("\t" + "\t" + '<LineStyle>' + "\r\n")
    f.write("\t" + "\t" + "\t" + '<color>50FFFFFF</color>' + "\r\n")
    f.write("\t" + "\t" + "\t" + '<width>3</width>' + "\r\n")
    f.write("\t" + "\t" + '</LineStyle>' + "\r\n")
```

```
        f.write("\t" + '</Style>' + "\r\n")
        f.write("\t" + '<Style id="lineStyle1">' + "\r\n")
        f.write("\t" + "\t" + '<LineStyle>' + "\r\n")
        f.write("\t" + "\t" + "\t" + '<color>50F0B414</color>' + "\r\n")
        f.write("\t" + "\t" + "\t" + '<width>3</width>' + "\r\n")
        f.write("\t" + "\t" + '</LineStyle>' + "\r\n")
        f.write("\t" + '</Style>' + "\r\n")
        #-------------------------------------------------------------------------#
        f.write("\t" + '<Style id="SubmergeStyler">' + "\r\n")
        f.write("\t" + "\t" + "<IconStyle>" + "\r\n")
        f.write("\t" + "\t" + "<Icon>" + "\r\n")
        f.write("\t" + "\t" + "<href>http://nettuno.ogs.trieste.it/sire/google_earth/glider/surfacing.png</href>" + "\r\n")
        f.write("\t" + "\t" + '</Icon>' + "\r\n")
        f.write("\t" + "\t" + '</IconStyle>' + "\r\n")
        f.write("\t" + '</Style>' + "\r\n")
        #-------------------------------------------------------------------------#
        f.write("\t" + '<Style id="GliderStyler">' + "\r\n")
        f.write("\t" + "\t" + '<IconStyle>' + "\r\n")
        f.write("\t" + "\t" + "\t" + '<Icon>' + "\r\n")
        f.write("\t" + "\t" + "\t" + "\t" + "<href>http://nettuno.ogs.trieste.it/sire/google_earth/glider/glider.png</href>" +
"\r\n")
        f.write("\t" + "\t" + "\t" + '</Icon>' + "\r\n")
        f.write("\t" + "\t" + '</IconStyle>' + "\r\n")
        f.write("\t" + '</Style>' + "\r\n")
        #-------------------------------------------------------------------------#
        f.write("\t" + '<Style id="NextStyler">' + "\r\n")
        f.write("\t" + "\t" + "<IconStyle>" + "\r\n")
        f.write("\t" + "\t" + "<Icon>" + "\r\n")
        f.write("\t" + "\t" + "<href>http://nettuno.ogs.trieste.it/sire/google_earth/glider/wpt.png</href>" + "\r\n")
        f.write("\t" + "\t" + '</Icon>' + "\r\n")
        f.write("\t" + "\t" + '</IconStyle>' + "\r\n")
        f.write("\t" + '</Style>' + "\r\n")
        f.write('<Folder>' + "\r\n")
        f.close()
        file_log = open(filelog, 'r')
        number_lines = len(file_log.readlines()) #trovo la lunghezza delle righe del file
        file_log.close()
        i = 1
        file_log = open(filelog, 'r')
        while 1:
```

```
line_or = file_log.readline()
if not line_or: break
line = line_or.split(" ")
if line[2] == "0":
        i = i + 1
        #---------estae campi data/ora---------#
else:
        data_fr_st = line[2]
        time_fr_st = line[3]
        data_st = line[6]
        time_st = line[7]
        data_end = line[10]
        time_end = line[11]
        dive_num = line[1]
        mission_num = line[0]
        #---------creazione struttura data/ora---------------#
        beg = "20" + data_st[4:6] + "-" + data_st[2:4] + "-" + data_st[0:2] + "T" + time_st[0:2] + ":"
+ time_st[2:4] + ":" + time_st[4:6] + "Z"
        end = "20" + data_end[4:6] + "-" + data_end[2:4] + "-" + data_end[0:2] + "T" +
time_end[0:2] + ":" + time_end[2:4] + ":" + time_end[4:6] + "Z"
        #----------------------lat/lon-----------------------#
        lat_deploy = line[5]
        lon_deploy = line[4]
        lat_beg = line[9]
        lon_beg = line[8]
        lat_end = line[13]
        lon_end = line[12]
        next_lat = line[24]
        next_lon = line[23]
        name = "sg554"
        if i == 1:
                if i == number_lines:
                        deploy = "20" + data_fr_st[4:6] + "-" + data_fr_st[2:4] + "-" +
data_fr_st[0:2] + "T" + time_fr_st[0:2] + ":" + time_fr_st[2:4] + ":" + time_fr_st[4:6] + "Z"
                        #---------------punto deploy-------------------#
                        head_desc_deploy = "deploy"
                        struttura(name, head_desc_deploy, mission_num, dive_num,
lat_deploy, lon_deploy, deploy, "#SubmergeStyler")

                        #---------------punto inizio-------------------#
                        head_desc = "first GPS"
```

```
                struttura(name, head_desc, mission_num, dive_num, lat_beg, lon_beg
,beg, "#SubmergeStyler")

                #----------------punto fine-----------------#
                head_desc = "last GPS"
                struttura(name, head_desc, mission_num, dive_num, lat_end, lon_end
,end, "#GliderStyler")

                #---------------prossimo punto----------------#
                head_desc = "target position"
                struttura_next(name, head_desc, next_lat, next_lon) #,estimated)
            else:
                deploy = "20" + data_fr_st[4:6] + "-" + data_fr_st[2:4] + "-" +
data_fr_st[0:2] + "T" + time_fr_st[0:2] + ":" + time_fr_st[2:4] + ":" + time_fr_st[4:6] + "Z"
                #---------------punto deploy----------------#
                head_desc_deploy = "deploy"
                struttura(name, head_desc_deploy, mission_num, dive_num,
lat_deploy, lon_deploy, deploy, "#SubmergeStyler")

                #---------------punto inizio----------------#
                head_desc = "first GPS"
                struttura(name, head_desc, mission_num, dive_num, lat_beg, lon_beg
,beg, "#SubmergeStyler")

                #----------------punto fine-----------------#
                head_desc = "last GPS"
                struttura(name, head_desc, mission_num, dive_num, lat_end, lon_end
,end, "#GliderStyler")

        elif i != 1:
            if i == number_lines:
                #---------------punto inizio----------------#
                head_desc = "first GPS"
                struttura(name, head_desc, mission_num, dive_num, lat_beg, lon_beg
,beg, "#SubmergeStyler")

                #----------------punto fine-----------------#
                #print "last i != 1 & i == number_lines"
                head_desc = "last position"
                struttura(name, head_desc, mission_num, dive_num, lat_end, lon_end
,end, "#GliderStyler")

                #---------------prossimo punto----------------#
```

```
                                head_desc = "target position"
                                struttura_next(name, head_desc, next_lat, next_lon)

                        else:
                                cost = 0
                                #---------------punto inizio------------------#
                                head_desc = "first GPS"
                                struttura(name, head_desc, mission_num, dive_num, lat_beg, lon_beg,
beg, "#SubmergeStyler")

                                #-----------------punto fine------------------#
                                head_desc = "last GPS"
                                struttura(name, head_desc, mission_num, dive_num, lat_end, lon_end,
end, "#SubmergeStyler")
                        i = i + 1
        file_log.close()
        file_log = open(filelog, 'r')
        while 1:
                line_or = file_log.readline()
                if not line_or: break
                line = line_or.split(" ")
                if line[2] == "0":
                        i = i + 1
                else:
                        lat_beg = line[9]
                        lon_beg = line[8]
                        lat_end = line[13]
                        lon_end = line[12]
                        data_fr_st = line[2]
                        time_fr_st = line[3]
                        data_st = line[6]
                        time_st = line[7]
                        data_end = line[10]
                        time_end = line[11]
                        dive_num = line[0]
                        mission_num = line[1]
                        first = "20" + data_fr_st[4:6] + "-" + data_fr_st[2:4] + "-" + data_fr_st[0:2] + "T" +
time_fr_st[0:2] + ":" + time_fr_st[2:4] + ":" + time_fr_st[4:6] + "Z"
                        beg = "20" + data_st[4:6] + "-" + data_st[2:4] + "-" + data_st[0:2] + "T" + time_st[0:2] + ":"
+ time_st[2:4] + ":" + time_st[4:6] + "Z"
```

```
                        end = "20" + data_end[4:6] + "-" + data_end[2:4] + "-" + data_end[0:2] + "T" +
time_end[0:2] + ":" + time_end[2:4] + ":" + time_end[4:6] + "Z"
                        next_lat = line[24]
                        next_lon = line[23]
                        gps1 = line[5] + "," + line[4] + "," + "0"
                        gps2 = line[9] + "," + line[8] + "," + "0"
                        gps_last = line[13] + "," + line[12] + "," + "0"
                        linestring("lineStyle", first, beg, gps1, gps2)
                        linestring("lineStyle1", beg, end, gps2, gps_last)
    f = open(file_kml, 'a')
    f.write('</Folder>' + "\r\n")
    f.write('</Document>' + "\r\n" + '</kml>' + "\r\n")
    file_log.close()
    f.close()



#--------------------------------- Main ---------------------------------

file_dati = "/storage/sire/work/web/sire/glider/dati_log.txt"
file_kml = "/storage/sire/work/web/sire/glider/sg554.kml"
if os.path.exists(file_kml):
    os.remove(file_kml)

crea_log_kml(file_kml, file_dati)
```