INTERNAL REPORT

Tracking in osservazioni spettroscopiche cometarie: la cometa C/2012 S1 (ISON)

Marco Buttu[(2)], Silvia Casu[(2)], Andrea Melis[(2)], Carlo Migoni[(2)],Sergio Poppi[(2)], Ignazio Porceddu[(2)] and the SRT Astrophysical ValidationTeam [(*)]

Report N. 41, released: 20/10/2014

Reviewer: Franco Buffa

OAC Osservatorio Astronomico di Cagliari

Roberto Ambrosini[1], Pietro Bolli[3], Marta Burgay[2], Paola Castangiai[2], Gianni Comoretto[3], Alessandro Corongiu[2], Nichi D'Amico[2], Elise Egron[2], Antonietta Fara[2,] Francesco Gaudiomonte[2],Federica Govonii[2], Noemi Iacolina[2], Matteo Murgiai[2], Francesco Nasyr[2], Alessandro Orfei[1], Andrea Orlati[1], Alberto Pellizzoni[2], Delphine Perrodin[2], Tonino Pisanu[2], Isabella Prandoni[1], Roberto Ricci[1], Simona Righini[1], Carlo Stanghellini[1], Andrea Tarchii[2], Caterina Tiburzi[2], Alessio Trois[2], Valentina Vacca[1],4, Giuseppe Valente[2], and Alessandra Zanichelli[1].

(1) INAF - Istituto di Radioastronomia, via P.Gobetti 101, 40129 Bologna, Italy.

(2) INAF - Osservatorio Astronomico di Cagliari, Loc. Poggio dei Pini, Strada 54, 09012 Capoterra (CA), Italy

(3) INAF - Osservatorio Astrofisico di Arcetri, Largo Enrico Fermi 5, I-50125, Firenze, Italy

(4) INAF - Dipartimento di Fisica e Astronomia, Università degli Studi di Bologna, Via Ranzani 1, 40127, Bologna, Italy

Tracking in osservazioni spettroscopiche cometarie: la cometa C/2012 S1 (ISON)

M.Buttu, S.Casu, A.Melis, C.Migoni, S.Poppi, I.Porceddu and the SRT Validation Team.

Abstract

Vengono di seguito presentate le strategie osservative adottate e l'architettura del programma sviluppato per effettuare un run osservativo TOO sulla cometa C/2012 S1 (ISON) nel periodo 25-28 novembre 2013.
Le osservazioni sono state effettuate in banda K in modalità spettroscopica, usando il backend XARCOS non ancora integrato nel sistema di controllo dell'antenna NURAGHE.

Contatti

Marco Buttu <mbuttu@oa-cagliari.inaf.it>
Silvia Casu <silvia@ira.inaf.it>
Andrea Melis <amelis@oa-cagliari.inaf.it>
Carlo Migoni <cmigoni@oa-cagliari.inaf.it>
Sergio Poppi <spoppi@oa-cagliari.inaf.it>
Ignazio Porceddu <porceddu@oa-cagliari.inaf.it>

Indice

1 Motivazioni scientifiche del TOO

Le comete sono corpi minori del Sistema Solare, il cui studio è di fondamentale importanza in quanto si ritiene che contengano materiale, per la maggior parte a base carbonio, proveniente dal disco protoplanetario solare da cui si sono originati i pianeti. Lo studio osservativo delle comete può dunque fornire utili indicazioni sulle prime fasi di formazione del Sistema Solare e sul contenuto organico primordiale. Le comete rappresentano inoltre una notevole sfida osservativa, in quanto corpi celesti che si muovono a considerevole velocità (centinaia di km/s) e quindi test fondamentali delle procedure di tracking preciso dei telescopi.

La cometa C/2012 S1 (ISON) è stata una cometa radente e non periodica, dall'orbita iperbolica molto inclinata rispetto all'eclittica. Di probabile provenienza dalla Nube di Oort, è stata scoperta nel 2012 [3]. Inizialmente annunciata come una cometa brillante, si è invece parzialmente disintegrata durante il suo passaggio al perielio, avvenuto il 28 novembre 2013.
La cometa ISON è stata oggetto di un run osservativo TOO (PI Porceddu) al Sardinia Radio Telescope (SRT) nel periodo 25-28 novembre 2013. Le osservazioni, effettuate con il backend XARCOS con il feed centrale, erano volte alla detection delle transizioni dell'ammoniaca (NH_3) in banda K (18-26 GHz). L'ammoniaca è una molecola top simmetrica che presenta un ricco spettro di roto-inversione, con numerose transizioni nel range spettrale 23-25 GHz, spesso usate come indicatori di temperatura. Molto probabilmente presente nella chioma e nei ghiacci cometari, è una molecola fondamentale per la comprensione della chimica dell'azoto nelle comete e, di conseguenza, per la caratterizzazione delle condizioni di formazione primordiale dei ghiacchi cometari.

La possibilità del backend spettroscopico XARCOS di usare più sottobande, permetteva di monitorare simultaneamente sia la transizione (1,1), avente frequenza a riposo di 23.694 GHz, sia la transizione (2,2), con frequenza a riposo di 23.723 GHz.

L'eventuale osservazione di ammoniaca nel nucleo cometario di ISON in preperielio e nell'immediato intorno temporale del perielio, benchè teoricamente molto difficile (l'ammoniaca dissocia facilmente e le sue transizioni sono in generale abbastanza deboli in comete non estremamente brillanti), avrebbe dato informazioni importantissime sulla composizione del nucleo e sulla chimica in atto.
Dal punto di vista dei test sulle modalità osservative di SRT, le osservazioni spettroscopiche di un oggetto in movimento veloce in banda K fornivano un'occasione unica per verificare le capacità di inseguimento e la stabilità del tracking di SRT, usando un servizio esterno di calcolo delle effemeridi.

2 Calcolo delle effemeridi

HORIZONS è il servizio per il calcolo di effemeridi di oggetti del sistema solare, sviluppato dal Jet Propulsion Laboratory(JPL) [5]. Tramite la sua interfaccia web

le effemeridi della cometa ISON sono state calcolate giornalmente, in modo specifico per il sito di SRT, e salvate su file, denominati file *horizons* nel seguito del report. Le posizioni della cometa ISON sono state tabulate con la risoluzione temporale di un minuto.

Come alternativa è stata considerata anche una libreria scritta in Python e chiamata *PyEphem* [4], la quale permette di effettuare il calcolo delle coordinate in tempo reale.

3 Strategia osservativa

La strategia *position switching* prevede l'alternarsi di un'osservazione puntando sulla sorgente (posizione **ON**), con una verso una posizione di riferimento fuori sorgente (**OFF**), ed è stata preferita ad altre strategie osservative, quali *nodding* e *frequency switching*, che avrebbero richiesto precursori non completati o non sufficientemente collaudati. Al momento delle osservazioni, il backend XARCOS non era ancora completemente integrato nel sistema di controllo dell'antenna, NURAGHE. Come primo test, si è comunque scelto di verificare la fattibilità dell'esecuzione di osservazioni via schedula. È stata dunque in primo luogo effettuata un'osservazione della transizione maser dell'H_2O a 22 GHz sulla sorgente W3(OH), utilizzando una schedula presistente di NURAGHE. Tuttavia la schedula non ha funzionato correttamente in quanto veniva saltato un subscan ogni due. Il problema, dovuto a diverse cause relative sia al component di XARCOS sia ad un non perfetto interfacciamento dello stesso con NURAGHE, è stato risolto e verrà descritto in un report in preparazione.

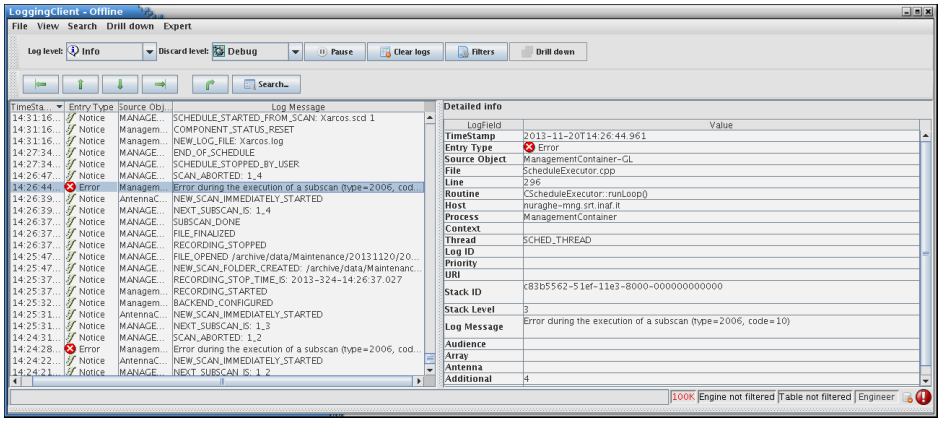

Figura 1: Schedula XArcos.scd, output di jlog

Vista l'impossibilità di effettuare una osservazione da schedula, nel modo routinario, si è scelto di implementare un programma ad hoc, che consentisse di comandare l'antenna e tutti i suoi dispositivi tramite l'intermediazione di NURAGHE.

Tale programma è completamente configurabile (vedi sezione 4.1) ed esegue una serie di cicli ON/OFF.

Per ciascuno di essi:

1. posiziona l'antenna *on source*
2. aspetta sinchè l'antenna è in tracking
3. imposta il valore dell'oscillatore locale
4. avvia l'acquisizione del backend e la scrittura dei dati su file
5. ferma l'acquisizione e la scrittura dei dati su file
6. posiziona l'antenna *off source*
7. aspetta sinchè l'antenna è in tracking
8. avvia l'acquisizione del backend e la scrittura dei dati su file
9. ferma l'acquisizione e la scrittura dei dati su file

Al termine del primo ciclo ON/OFF, e dopo un certo numero di cicli, quando l'antenna è *off source* il programma effettua una calibrazione, ovvero:

- attiva il generatore di marca di rumore
- avvia l'acquisizione del backend e la scrittura dei dati su file
- ferma l'acquisizione e la scrittura dei dati su file
- disattiva il generatore di marca di rumore

I tempi di acquisizione *on source*, *off source* e durante una calibrazione sono uguali. Il posizionamento dell'antenna, come vedremo nella sezione 5, viene eseguito in modo concorrente rispetto agli altri task.

4 Architettura del programma

Il programma, scritto nel linguaggio di programmazione Python, è volutamente di uso generale in quanto consente di effettuare una osservazione ON/OFF, così come descritta nella sezione 3, a prescindere dalla sorgente, dal sito osservativo (SRT, Medicina, ecc.) e dai parametri tecnici dell'osservazione.

4.1 Parametri di configurazione

Il programma è configurabile sulla base dei seguenti parametri (tra parentesi sono indicati i valori di default):

- attivazione modalità **simulazione**: l'osservazione viene simulata, nel senso che il programma viene eseguito allo stesso modo che nel caso di una osservazione reale, calcolando le posizioni, i valori del LO ecc., ma quando arriva il momento di comandare i dispositivi d'antenna, l'invio dei comandi viene saltato e ci si limita a salvare i log e mostrare a video i messaggi di avvenuta esecuzione
- numero di **cicli** ON/OFF da compiere (10)
- percentuale di cicli di **calibrazione** da compiere (10%)
- durata di ciascuna **acquisizione** (on source, off source e di calibrazione) da parte del backend (60 secondi)
- tempo che intercorre tra due **posizionamenti** successivi dell'antenna (5 secondi)
- nome del **sito osservativo**: SRT, Medicina, ecc. Sulla base di questo vengono caricate le informazioni relative al sito (latitudine, longitudine, altitudine, nome completo)
- parametri orbitali della **sorgente**
- **offset** da applicare alla posizione della sorgente, per andare off source
- riferimento per il calcolo del tempo dell'**oscillatore locale**: 0.5 significa che il valore del LO viene calcolato al centro dell'intervallo di acquisizione on source, 0 significa che viene calcolato all'inizio dell'acquisizione, 1 alla fine, ecc. (0.5 per default)
- nome del file **horizons**: da questo file vengono reperiti la velocità radiale e la posizione della sorgente; per maggiori dettagli si veda la Sezione 5
- **frequenza di laboratorio** della transizione (default 23694.5 MHz: ammoniaca (1,1))
- **frequenza inferiore** della banda del backend (default 155.0 MHz)
- **frequenza superiore** della banda del backend (default 170.6 MHz)

Le osservazioni sono state condotte utilizzando i valori riportati tra parentesi, a parte per il giorno 28 Novembre in cui si è passati ad osservare l'acqua (frequenza di laboratorio di 22235.08 MHz) piuttosto che l'ammoniaca.

4.2 Struttura

Il programma è strutturato in 5 moduli:

1. `onoff_switching.py`: è il main file da eseguire per avviare il programma; contiene i valori dei parametri dell'osservazione ed effettua il parsing degli argomenti passati da linea di comando
2. `devices.py`: esporta i riferimenti ai dispositivi di antenna (mount, antenna, ricevitore, backend, LO)
3. `handler.py`: è lo scheduler che gestisce l'esecuzione dei task durante l'osservazione
4. `models.py`: definisce i modelli da utilizzare per la sorgente e per il sito osservativo
5. `config.py`: effettua la validazione dei parametri di configurazione.

5 Posizionamento dell'antenna

Il posizionamento dell'antenna viene effettuato da un task (un processo daemon), chiamato *posizionatore*, che opera in modo concorrente rispetto alle restanti attività (acquisizione, settaggio LO, ecc.) Lo scheduler comunica al posizionatore quando andare off source e quando on source.

Le posizioni da comandare (a partire dal tempo attuale) vengono lette dal file *horizons*, e caricate in memoria all'avvio del programma. All'atto del posizionamento, la posizione da comandare viene calcolata effettuando una iterpolazione lineare tra i valori con marca temporale immediatamente inferiore e superiore al tempo attuale (la risoluzione temporale del file horizons era di un minuto). La posizione da comandare viene poi confrontata con quella ottenuta utilizzando la libreria PyEphem, alla quale vengono passati i parametri orbitali della sorgente. Questo confronto è utile per verificare che non siano stati commessi errori nel calcolo della posizione. Nel caso tale differenza di posizione sia superiore al beam, viene mostrato e messo a log un messaggio di warning[1].

I valori `ra`, `dec` vengono comandati utilizzando il metodo `sidereal()` del component `AntennaBoss` di Nuraghe, nel seguente modo:

```
antenna.sidereal(ra, dec, ANT_J2000, ACU_NEUTRAL)
```

[1]Si è osservato che le posizioni ottenute con PyEphem differivano da quelle ottenute con il file horizons, e tali differenze aumentavano con il passare dei giorni. Il fenomeno è spiegabile con il fatto che i parametri orbitali di PyEphem erano aggiornati al 4 novembre mentre il file *horizons* veniva creato quotidianamente.

6 Tracking

Il tracking è stato gestito controllando la posizione del `Mount`, e verificando che la distanza tra la posizione comandata e quella attuale fosse inferiore ad un decimo di beam ($\sim 2 \cdot 10^{-4} deg.$)

```
from math import *
distance = sqrt(abs(cmd_az - act_az)**2 + abs(cmd_el - act_el)**2)
track = True if distance < 2e-5 else False
```

Dopo che l'antenna è considerata in tracking passano comunque 2 secondi prima dell'acquisizione (tempo necessario affinchè il backend sia pronto per acquisire).

7 Oscillatore locale

Il valore del LO viene calcolato sulla base del valore della velocità radiale della sorgente, letto dal file horizons. Il valore della velocità radiale è ottenuto effettuando una iterpolazione lineare tra i valori con marca temporale immediatamente inferiore e superiore al tempo centrale dell'acquisizione. La formula utilizzata per il calcolo del valore dell'oscillatore locale è la seguente:

```
vobs = lab_freq * (1 - radialSpeed/c)
lo = vobs - (backend_lower_freq + backend_upper_freq) / 2
```

Il LO viene impostato subito dopo che l'antenna è considerata in tracking, e prima che inizi l'acquisizione. Dal momento in cui viene impostato il LO a quello in cui l'acquisizione parte intercorrono circa due secondi, necessari per inizializzare il backend. Questo tempo è tenuto in conto nel calcolo del tempo di riferimento del LO.

7.1 Risoluzione bug oscillatore locale

Il sistema di monitoraggio dei ricevitori controlla periodicamente il flag *isLocked* fornito dalla property del component di controllo degli oscillatori locali, che, piú precisamente, confronta il valore comandato dal component con quello effettivamente impostato nel sintetizzatore; se i due valori discordano viene segnalato un *failure* nel sottosistema ricevitori.

In particolare, per questo esperimento era necessario impostare il valore con una risoluzione dell'ordine della decina di kHz. Un errore nella conversione del valore da virgola mobile a stringa, come richiesto dal linguaggio `SCPI` (Standard Commands for Programmable Instruments), che causava un troncamento alla prima cifra decimale del valore in MHz, creava una discordanza con il valore comandato. L'errore è stato individuato nella classe di conversione da *float* a *string* in quanto

nella classe seguente, senza specificare il *flag* di formato nello stream, la scrittura su stringa avveniva senza le cifre decimali necessarie:

```
inline string stringify(double x)
 {
   std::ostringstream o;
   if (!(o << x))
     throw BadConversion("stringify(double)");
   return o.str();
 }
```

Nella classe successiva, invece, si è utilizzato uno specificatore di formato in modo da scrivere un numero significativo di cifre decimali sufficiente per le esigenze:

```
inline string stringify(double x)
 {
   std::ostringstream o;
   if (!(o <<fixed<<setw(8) << x))
     throw BadConversion("stringify(double)");
   return o.str();
 }
```

Un'ulteriore causa di errore è stata riscontrata nel confronto tra i valori in quanto avveniva con un'uguaglianza tra float, pertanto possibile causa di errore a causa della modalità di rappresentazione dei valori in cifra mobile all'interno della memoria del computer. Pertanto si è optato per un confronto affinché la differenza tra valore comandato e valore letto rimanga entro una soglia predefinita:

```
if (fabs(expected_freq - read_frequency) <= SYNTH_TOLLERANCE)
```

8 Backend

Il backend utilizzato per l'osservazione è XARCOS, uno spettropolarimetro in grado di analizzare in contemporanea fino a 16 segnali con una banda di 125 MHz ciascuno. Poichè l'osservazione è stata condotta con il solo beam centrale del ricevitore in banda K, si sono utilizzate solamente le due IF corrispondenti.

Tra le peculiarità di XARCOS c'è anche quella di poter processare la stessa IF con più spettrometri; ciò ci ha consentito di analizzare due differenti porzioni di banda del beam, permettendoci di osservare entrambe le linee (1,1) e (2,2) dell'ammoniaca, le cui frequenze di riposo sono rispettivamente 23.694 e 23.723 GHz.

Non essendo richieste analisi relative alle cross-correlazioni fra le due polarizzazioni, le informazioni corrispondenti, pur risultando presenti nei file prodotti, devono essere ignorate.

Il software osservativo (un component ACS chiamato `XBackends`) con cui viene gestito il backend XARCOS tramite NURAGHE era completo ma non ancora perfettamente funzionante. Il component talvolta rimaneva in uno stato di busy, e non si riusciva più ad effettuare delle acquisizioni. Altre volte quando si tentava di ottenere il riferimento al component, questo andava in segmentation fault. Altre volte ancora, il malfunzionamento del component di XARCOS causava il crash del component `FitsZilla`, il quale si occupa di leggere i dati dal backend e scriverli su file in formato FITS.

Tali problematiche ovviamente impedivano l'utilizzo del component di XARCOS da schedula NURAGHE durante una osservazione, e questo è il principale motivo che ci ha spinti a scegliere una soluzione custom.

La configurazione finale del sistema, soddisfacente sotto molti aspetti, è stata individuata dopo varie prove nel modo seguente:

```
class Recorder(object):
    def __init__(self, lower_freq, upper_freq,
            backend_name='BACKENDS/XBackends',
            writer_name='MANAGEMENT/FitsZilla',
            projectName='ison', observerName='Gavino'):
        client = PySimpleClient()
        self.writer = client.getComponent(writer_name)
        self.backend = client.getComponent(backend_name)
        self.backend.setSectionsNumber(2)
        self.backend.setSection(0, lower_freq - 125,
                upper_freq - lower_freq, 1, 2, 31.25, -1)
        self.backend.setSection(1, 60,
                upper_freq - lower_freq, 1, 2, 31.25, -1)
                self.abs_obs_path = join(abspath(curdir), obs_datadir)
        # [CUT] output directory creation...
        self.scanSetup = Management.TScanSetup(
            scanTag=0,
            scanId=1,
            projectName=projectName,
            observerName=observerName,
            path=self.abs_obs_path,
            extraPath='',
            baseName='',
            scanLayout='',
            schedule='noSchedule',
            device=0)
        self.subScanSetup = Management.TSubScanSetup(
            startUt = getTimeStamp().value,
            subScanId=0,
```

```
            axis=Management.MNG_NO_AXIS,
            targetID='',
            extraPath='',
            baseName='')
        self.startDelay = 2.0 # Seconds before the acquistion starts

    def start(self, cycle, cycle_type):
        self.writer.reset();
        time.sleep(self.startDelay/2)
        self.backend.connect(self.writer);
        self._updateScanSetup(cycle, cycle_type)
        self.writer.startScan(self.scanSetup)
        time.sleep(self.startDelay/2)
        self.backend.sendHeader()
        self._updateSubScanSetup()
        self.writer.startSubScan(self.subScanSetup);
        self.backend.sendData(self.subScanSetup.startUt)
        # Wait at least 11 seconds
        while getTimeStamp().value < \
                self.subScanSetup.startUt + 110000000:
            time.sleep(1)

    def stop(self):
        self.backend.sendStop();
        while self.writer.isRecording():
            time.sleep(1)
        time.sleep(2)
        self.writer.stopScan()
        time.sleep(1)
        self.backend.terminate()
        self.backend.disconnect()
        self.writer.closeReceiver()

    def _updateScanSetup(self, cycle, cycle_type):
        # [CUT] cycle_type_datadir creation...
        self.scanSetup.scanTag += 1
        self.scanSetup.path = abspath(cycle_type_datadir)

    def _updateSubScanSetup(self):
        # Add one second (10000000)
        self.subScanSetup.startUt = getTimeStamp().value + 10000000
        self.subScanSetup.subScanId += 1
```

Come si può notare, all'inizio dell'osservazione, quando viene creata l'istanza della

classe `Recorder`, vengono richiesti i component di XARCOS e del `FitsWriter`, e poi vengono create le configurazioni di setup dello scan e del sub-scan, che verranno poi passate al `FitsWriter` prima di ciascuna acquisizione. Nel metodo `Recorder.stop()`, oltre ad effettuare un `writer.stopScan()` e un `backend.terminate()`, vengono chiamati in aggiunta i metodo `backend.disconnect()` e `writer.closeReceiver()`. Queste ultime due chiamate consentono di evitare sia che XARCOS vada in stato di busy, sia che il `FitsWriter` cessi di funzionare correttamente. Per contro, viene creato un file FITS per ogni acquisizione.

8.1 Scrittura dei file FITS e architettura della directory

I file FITS prodotti sono stati salvati a partire dalla seguente directory della macchina `nuraghe-mng`:

```
cd ~/Nuraghe/ACS/trunk/Common/Misc/OnOffSwitching/ison_data/
```

Per capire come è articolata la struttura di directory, consideriamo come esempio i dati relativi alla sessione osservativa del 26 novembre:

```
ls -R 2013-Nov-26/
2013-Nov-26/:
001
002
003
...
```

Per ogni osservazione, viene creata una directory numerata in ordine progressivo, a partire da 001. Ciascuna di queste directory contiene:

1. un file *conf.txt* contenente i dati relativi all'osservazione,
2. un file *short_log.txt* contenente un sommario dei log,
3. un file con estensione *.log*, contenente l'elenco competo dei log,
4. una serie di directory *cycle**, ciascuna delle quali contiene dati relativi ad un ciclo ON/OFF:

```
2013-Nov-26/001:
conf.txt
cycle01
cycle02
cycle03
cycle04
cycle05
cycle06
```

```
cycle07
cycle08
cycle09
cycle10
ison--2013_11_26-06_48_17.log
short_log.txt
```

La directory *cycle01* contiene ad esempio tre sottodirectory:

```
2013-Nov-26/001/cycle01:
calibration-h06m51s03
off_source-h06m49s49
on_source-h06m48s28
```

Ciascuna di esse contiene il relativo file FITS:

```
2013-Nov-26/001/cycle01/calibration-h06m51s03:
_001_003.fits

2013-Nov-26/001/cycle01/off_source-h06m49s49:
_001_002.fits

2013-Nov-26/001/cycle01/on_source-h06m48s28:
_001_001.fits
```

Per ogni giornata di osservazioni è stata creata un directory (2013-Nov-26, 2013-Nov-27, 2013-Nov-28, 2013-Nov-29), secondo l'architettura sopra rappresentata:

```
-- 2013-Nov-26
|   |-- 001
|   |   |-- conf.txt
|   |   |-- cycle01
|   |   |   |-- calibration-h06m51s03
|   |   |   |   `-- _001_003.fits
|   |   |   |-- off_source-h06m49s49
|   |   |   |   `-- _001_002.fits
|   |   |   `-- on_source-h06m48s28
|   |   |       `-- _001_001.fits
|   |   |-- cycle02
|   |   |   |-- off_source-h06m53s45
|   |   |   |   `-- _001_005.fits
|   |   |   `-- on_source-h06m52s22
|   |   |       `-- _001_004.fits
|   |   |-- cycle03
|   |   |   |-- off_source-h06m56s27
```

```
|   |   |   |   `-- _001_007.fits
|   |   |   `-- on_source-h06m55s06
|   |   |       `-- _001_006.fits
|   |   |-- cycle04
|   |   |   |-- off_source-h06m59s16
|   |   |   |   `-- _001_009.fits
|   |   |   `-- on_source-h06m57s52
|   |   |       `-- _001_008.fits
|   |   |-- cycle05
|   |   |   |-- off_source-h07m01s59
|   |   |   |   `-- _001_011.fits
|   |   |   `-- on_source-h07m00s39
|   |   |       `-- _001_010.fits
|   |   |-- cycle06
|   |   |   |-- off_source-h07m04s45
|   |   |   |   `-- _001_013.fits
|   |   |   `-- on_source-h07m03s21
|   |   |       `-- _001_012.fits
|   |   |-- cycle07
|   |   |   |-- off_source-h07m07s28
|   |   |   |   `-- _001_015.fits
|   |   |   `-- on_source-h07m06s06
|   |   |       `-- _001_014.fits
|   |   |-- cycle08
|   |   |   |-- off_source-h07m10s15
|   |   |   |   `-- _001_017.fits
|   |   |   `-- on_source-h07m08s53
|   |   |       `-- _001_016.fits
|   |   |-- cycle09
|   |   |   |-- off_source-h07m13s03
|   |   |   |   `-- _001_019.fits
|   |   |   `-- on_source-h07m11s38
|   |   |       `-- _001_018.fits
|   |   |-- cycle10
|   |   |   |-- off_source-h07m15s51
|   |   |   |   `-- _001_021.fits
|   |   |   `-- on_source-h07m14s27
|   |   |       `-- _001_020.fits
|   |   |-- ison--2013_11_26-06_48_17.log
|   |   `-- short_log.txt
|   |-- 002
|   |   |-- conf.txt
|   |   |-- cycle01
```

```
|   |   |   |-- calibration-h07m27s25
```

9 Analisi acquisizioni

XARCOS salva i dati in blocchi da 10 s di integrazione, denominati *dump*. Pertanto, per ogni file ci si aspetta un numero di dump pari al rapporto tra il tempo di acquisizione desiderato e la durata del dump. Quando si invia uno stop, XARCOS continua ad acquisire sinchè non completa il dump di 10 secondi. Si è osservato però che se non sono ancora trascorsi i primi 10 secondi dall'inizio dell'acquisizione, il component di XARCOS resta in uno stato di busy. Per questo motivo, si è deciso di forzare con uno sleep di 11 secondi l'operazione di start.

Inoltre, si è osservato che viene salvato nel file FITS un numero di dump inferiore a quello che ci si aspetta, e questo numero dipende dal tempo di acquisizione. Ad esempio, con un tempo di acquisizione di 40 secondi, si ottengono 3 dump da 10 secondi per il primo FITS, e per i FITS successivi un numero di dump che varia tra 2 e 4. Con un tempo di 60 secondi si ottengono 5 dump per il primo fits e 6 dump per i restanti. Si è deciso quindi di utilizzare un tempo di acquisizione di 60 secondi[2]. Per verificare questo comportamento sono stati analizzati alcuni file relativi ad acquisizioni ON, OFF e CAL dell'acquisizione 2013-Nov-27/001.

In figura 2 si può verificare che, sebbene i file avessero tutti lo stesso tempo di acquisizione, in un caso (cycle 01 on) sono stati registrati 5 dump, mentre negli altri compaiono 6 dump.

Analizzando il contenuto dei dump nei vari cicli, si possono individuare però alcune irregolarità.

In figura 2, il plot in basso a destra è relativo ad un'acquisizione con la marca di calibrazione accesa (CAL), tuttavia si può notare che il dump 0 ha un livello più basso degli altri dump, pari al livello di una posizione OFF. Analogamente, nel plot in basso a destra, corrispondente alla posizione ON del ciclo 2 mostra che il dump 0 ha un livello più elevato, corrispondente a quello di un'acquisizione di calibrazione. Poiché la sequenza di acquisizioni è la seguente:

- Ciclo 1
  - ON;
  - OFF;
  - CAL;
- Ciclo 2
  - ON;
  - OFF;

[2]Si comanda uno stop ad XARCOS dopo 62 secondi.

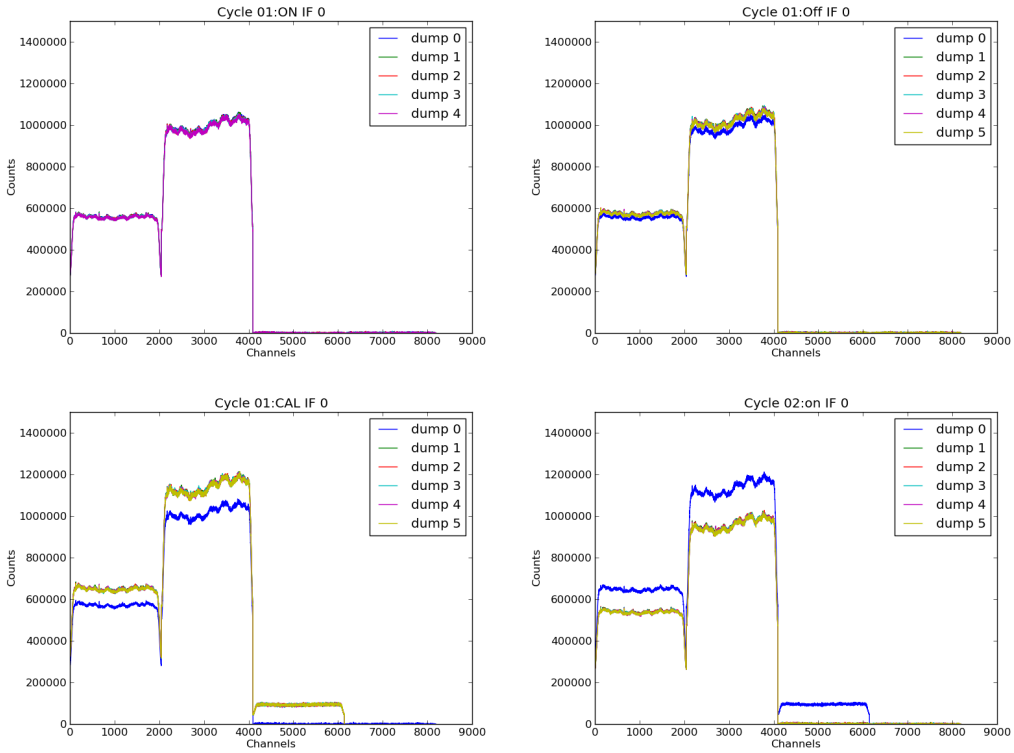

Figura 2: Analisi *dump*: cycle 01 on source (in alto a sinistra), cycle 01 off source (in alto a destra), cycle 01 calibrazione (in basso a sinistra), cycle 02 on source (in basso a destra).

- Ciclo 3
  - ON;
  - OFF;
- Ciclo 4
  - ...

è ipotizzabile che il dump 0 della posizione CAL sia invece riferibile alla posizione precedente, ovvero OFF. Analogamente, il primo dump della posizione ON del secondo ciclo sembra riferirsi alla precedente posizione CAL.

Riassumendo, ciascun dump, tranne nel primo ciclo, sembra riferirsi ad una posizione precedente nella sequenza ON-OFF-[CAL]. Pertanto, durante l'analisi dei dati si dovrà tenere conto del fatto che:

- il primo FITS può essere analizzato così come è

- per i FITS successivi, il primo dump probabilmente è relativo alla acquisizione precedente per cui si consiglia di scartare il primo dump di ciascun FITS successivo al primo.

Successivamente, si è scoperto che il problema era legato al container FitsZilla, che finalizzava il file FITS mentre XARCOS stava ancora processando l'ultimo dump. In pratica, FitsZilla manteneva in memoria il dump per poi andarlo a scrivere nel FITS successivo, generando il problema descritto. I dettagli relativi alla soluzione del problema verranno illustrati in un technical report in preparaizone.

10 Validazione del sistema

Prima di effetturare le osservazioni vere e proprie sulla cometa ISON, al fine di verificare il corretto funzionamento di tutto il sistema, si è scelto di osservare una sorgente di forte emissione nelle transizioni osservate. La scelta della sorgente da usare come test del sistema è ricaduta sulla dark cloud **L483** (RA=$18^h17^m35^s$, DEC -04°39′.8, J2000, V_{LSR}=+5.460 Km/s) che possiede una forte emissione nella transizione NH_3 (J,K)=(1,1)[2] e si trovava, tra l'altro, in condizioni di visibilità ottimali.

Per inviare le coordinate di L483 allo scheduler è stato modificato il file delle effemeridi sostituendo le coordinate della cometa con quelle della sorgente di test. Analogamente, è stata modificata la velocità radiale dell'oggetto. I dati di test sono reperibili a partire dalla directory **013-Nov-27/010**. In figura 3 è rappresentato il risultato dell'acquisizione, dopo aver sommato spettri **ON-OFF** di 10 cicli, corrispondenti a 500 secondi di integrazione.

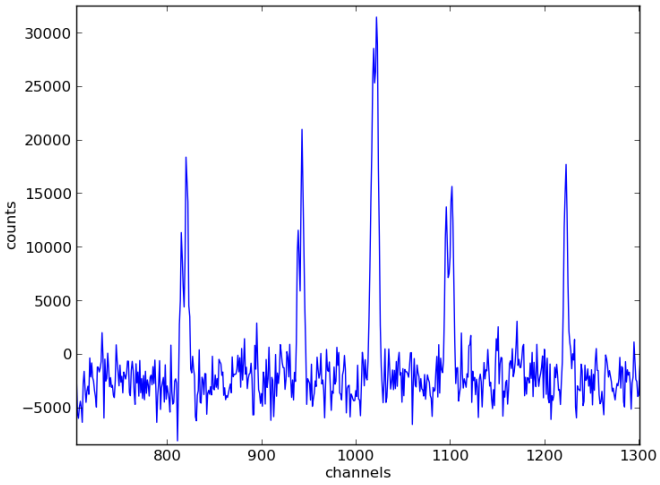

Figura 3: L483:ON-OFF, 500 secondi on source

Dal confronto con il profilo teorico[1], si può evincere un corretto funzionamento del sistema.

Riferimenti bibliografici

[1] G. Anglada, I. Sepulveda, and J. F. Gomez. Ammonia observations towards molecular and optical outflows. , 121:255–274, February 1997.

[2] G. A. Fuller and P. C. Myers. Thermal Material in Dense Cores: A New Narrow-Line Probe and Technique of Temperature Determination. *ApJ*, 418:273, November 1993.

[3] V. Nevski and E. Romas. Comet Observations [D00 ISON-Kislovodsk Observatory]. *Minor Planet Circulars*, 81571:34, December 2012.

[4] PyEphem. http://rhodesmill.org/pyephem/.

[5] HORIZONS System. http://ssd.jpl.nasa.gov/?horizons.