The GIANO control software system

Emanuel Rossetti* [a], Ernesto Oliva[b,c], Livia Origlia[a]

[a]INAF, Osservatorio Astronomico di Bologna, Via Ranzani 1, I-40127 Bologna, Italy;
[b]INAF, Osservatorio Astrofisico di Arcetri, Largo E. Fermi 5, I-50125 Firenze, Italy;
[c]INAF, Telescopio Nazionale Galileo, P.O. Box 565, Santa Cruz de La Palma, TF, Spain.

ABSTRACT

GIANO is an ultra-stable IR echelle spectrometer, optimized for both low (R $\simeq$ 400) and high (R $\simeq$ 50,000) resolution, that will be installed at the Nasmyth-B focus of the Italian national telescope (TNG).
At the beginning of this year the assembling phase of GIANO has started, at the Infrared Laboratory of INAF-Arcetri, and is currently in progress.
We describe, here, the general control software structure of the instrument concerning both the user interface and the controls of all subsystems. We present also the software interface which provides the communication with the cryogenic system of the instrument and is handled by means of a Programmable Logic Controller.

Keywords: Control software, PLC, infrared spectrometer.

1. INTRODUCTION

GIANO is a cross-dispersed high resolution infrared spectometer that will be installed at the Nasmyth-B focus of the Telescopio Nazionale Galieo (TNG). The TNG is a 3.58m (F/11) alto-azimuthal telescope located at Roque de Los Muchachos Observatory (ORM), La Palma, Spain equipped with an active optics system (detailed information on TNG can be found at *http:://www.tng.iac.es*).
GIANO has been devised to perform spectroscopic observations on a wide wavelength range (from 0.9μm to 2.5μm) both at low ($\simeq$400) and high ($\simeq$50,000) resolution, configurations that can be achieved combining a set of prisms with an echelle grating.
The primary aim of GIANO is to achieve the highest possible image quality and spectral stability as they are essential conditions for precise radial velocity measurements. Further details on the instrument and on its performances can be found in Oliva et al.[1] (2006).

In latter end of 2007 the Criotec Impianti S.r.l. (a company specialized in cryogenic constructions) completed the manufacturing of the GIANO cryostat and of its optical bench. After the testing phase, the overall cryogenic system was delivered to the Infrared Laboratory of INAF Osservatorio Astrofisico di Arcetri where, at the beginning of 2008, the mounting of various spectrometer components started. As soon as the mounting of the motors on filter and slit wheels was accomplished we tested the control software modules for GIANO cold movements which turned out to be efficient, fast and reliable.

The decision to make the GIANO cryogenic system to be controlled by an industrial PLC (Programmable Logic Controller), which communicates, at low level software, with the GIANO workstation, imposed the redesign of a part of the GIANO Instrument Control Software and the writing of a few new software modules devoted to the interface with the PLC.
In this paper we give a brief description of the GIANO control software and a few further details concerning the cryogenic telemetry acquisition.

* Further author information: emanuel.rossetti@oabo.inaf.it

Advanced Software and Control for Astronomy II, edited by Alan Bridger, Nicole M. Radziwill
Proc. of SPIE Vol. 7019, 70191S, (2008) · 0277-786X/08/$18 · doi: 10.1117/12.788978

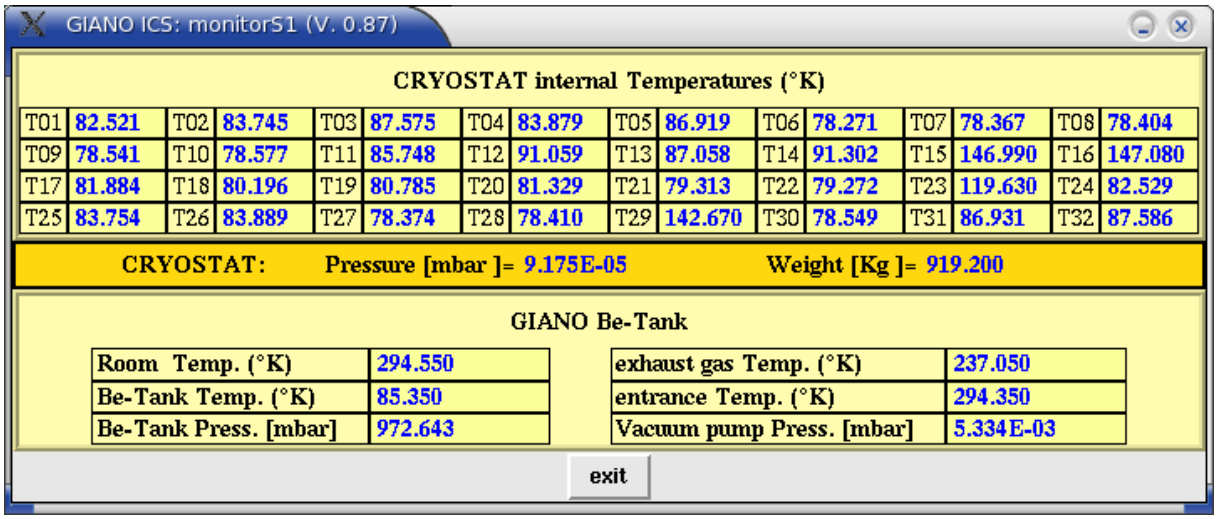

Figure 1. The GIANO IS1 main panel during a cooling session.

2. THE INSTRUMENT CONTROL SOFTWARE

The GIANO instrument control software (ICS) is based on a classical software scheme already described in [4] and follows a modular structure. The GIANO ICS runs on a dedicated PC-Linux workstation and besides performing software control of the whole system is also responsible for the control of all sensors and of moving mechanisms, both inside and outside the GIANO cryostat. The management of all these controls is handled by dedicated tasks called *Instrument Status* (IS).
Almost all ICS modules are written in Tcl/Tk scripting language and the communications between the GIANO ICS and all IS tasks are performed by means of IP or local sockets.
The ICS is also interfaced with the GIANO DCS (Detector Control Software). The latter is structured in two different parts: one residing inside the PC-Linux and the other running on the embedded processor at the focal plane electronics and having direct access to the hardware. These two distinct branches are interfaced by means of a standard *ethernet* connection through which data and commands are sent to Unix-like sockets. The software portion inside the embedded processor manages the section of the appropriate waveform for the detector, controls the data flow and can handle special observing modes such as multiple starts and stops. Further details about GIANO DCS can be found in Baffa et al.[3]

2.1 Temperature sensors

GIANO has 32 temperature sensors positioned within the cryostat that are used to monitor the temporal and spatial variation of the temperature during the instrument cooling/warming process and during the normal operations. Up to 8 temperature sensors can be connected to a commercial *Lake Shore 218* temperature monitor which is serially interfaced with a *Lantronix* terminal server. As a whole there are 4 *Lake Shore* temperature monitors always on.
At software level the monitoring of all temperature sensors is performed, without interruptions, on reading cycles of ∼10 seconds by `IS1temp` (a daemon always running in the background). At start-up this task needs to read a configuration file containing both the *Lantronix* IP address and port numbers for each connected *Lake Shore*. Since the configuration file includes also a list of the connected temperature sensors it is possible to modify it to exclude, from the reading cycles done by `IS1temp`, one or more *Lake Shore*. All available/read temperature values are stored in a local rotational data file and are sent, on reading, through an IP socket, to the TNG database (if available/on-line). All these temperatures can be displayed on the GIANO workstation by launching the IS1 GUI (Fig.1 upper) which shows also (Fig.1 lower) a few crucial cryostat telemetry parameters (see section 2.2). This panel is automatically refreshed after new reading done by `IS1temp`.

Figure 2. The GIANO BeTank.

2.2 PLC controls

The core structure of the GIANO cryostat is an aluminum optical bench and a liquid nitrogen tank (called *BeTank*, see Gennari et al.[2]) allowing a large (up to $\simeq$100 liters) reserve of liquid nitrogen at the very center of the cryostat and in direct contact with the surface of the optical bench (see Fig.2).
The vacuum and cryogenic devices are controlled by an industrial PLC which performs the following basic actions:

- management of all the vacuum tubes pumps and valves;
- control of the liquid nitrogen flux entering the cryostat;
- control of the fluxes and pressures of nitrogen getting off the cryostat;
- management of the heaters for the fast heating of the system.

For safety reasons not all the components of the cryogenic system can be activated via software. For example both the vacuum and the turbo pumps can be switched on and off only manually. The PLC records, however, the status of these pumps which can be displayed with the PLC telemetry panel (see Fig.3). This latter can be launched to view and/or modify the values of a series of variables. The monitoring of all cryostat telemetry variables is performed without interruption with reading cycles of $\sim$10 seconds by `IS1plc` (a daemon always running in the background). This daemon needs a configuration file at start-up level and writes in a rotational log file all telemetry values got from the GIANO PLC. Moreover it sends the telemetry data to the TNG database

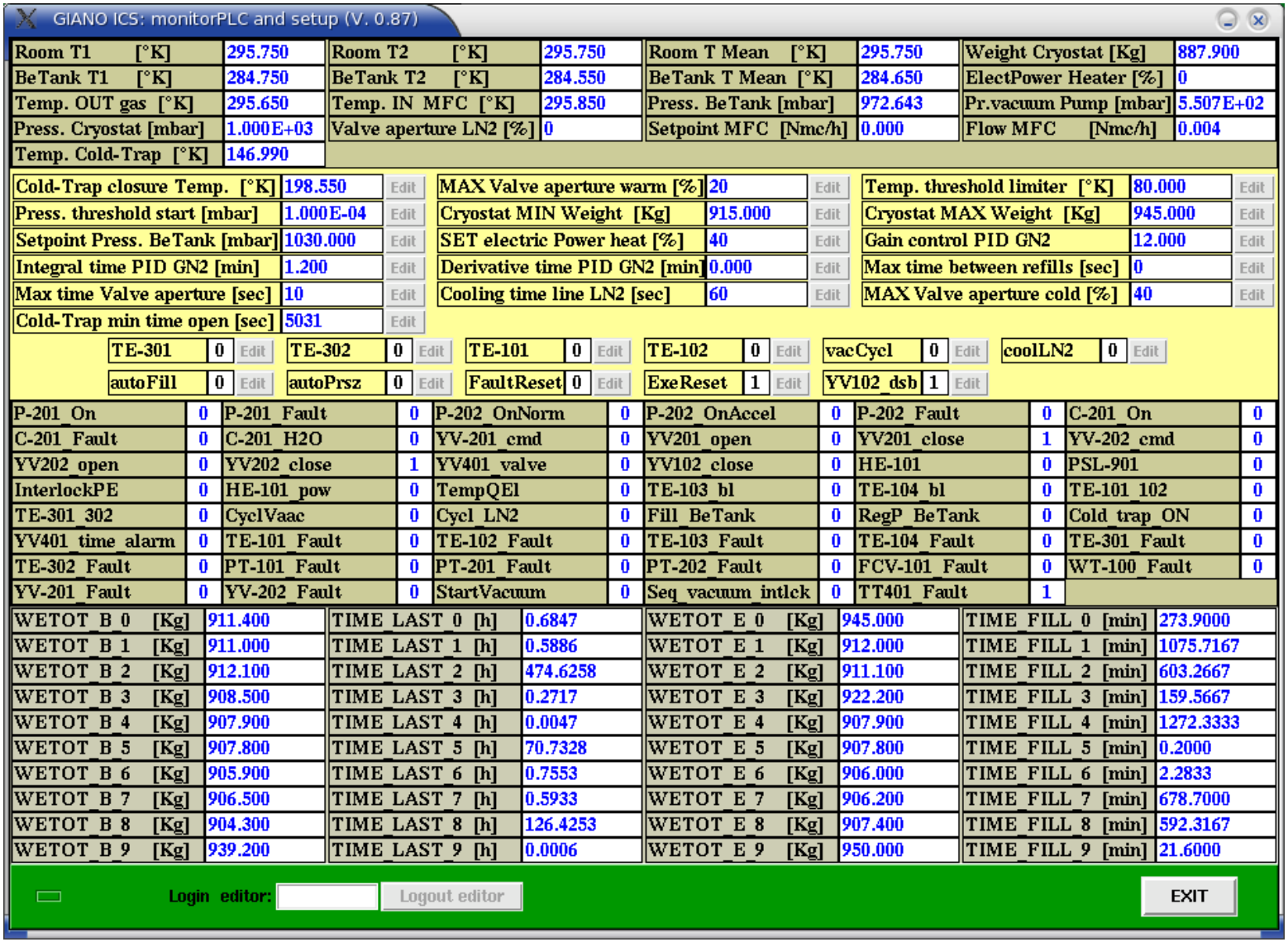

Figure 3. GIANO cryostat PLC full telemetry panel.

(if it is on-line) through an IP socket. If the telemetry panel is kept open, its values are updated after each new reading performed by `IS1plc`.

The PLC communicates with the GIANO PC-Linux through a RS232 serial line which is re-routed on the *ethernet* by a *Lantronix* terminal server. The communication protocol used is the "Modbus RTU" with binary encoding of data and a 16 bit CRC check for detection of transmission errors. The cryostat telemetry variables (32-bit float values) are stored as pairs of two consecutive 16-bit registers (words), while the cryostat logical variables make use of 1-bit and are stored in single word registers.

The GIANO PLC has been programmed to store all the cryostat data information in 150 (words) registers. The floating variables which have been defined and are available for the Modbus communication are 73 and occupy 146 registers. Most (57) of these variables are only readable, the remaining 16, instead, are also writable and are displayed in Fig.3 (yellow frame). It is possible to modify the values of these 16 variables acting on each corresponding "edit" button which opens a little window in which the new value can be inserted. The new value will be sent to PLC and updated on the GUI provided that it is inside the possible range (also displayed in the little window). The remaining 4 registers (of which only 1 is writable) are reserved and used to store/set only logical information (single bits) like the opening/closing of valves, the fault reset, the signal of *BeTank* refilling, the fault of sensors, the alarms and so on.
In the same yellow area of Fig.3 other editable variables are visible. In these cases only one value (logical) can be inserted. These variables relate to reset and opening/closing of valves which can be handled via software. Of course these modifying options are possible only in super-user mode (password is required).

The data displayed in the telemetry panel can be graphically represented by means of `IS1plot` tool which provides visual real time monitoring (strip chart) of the variation of GIANO cryogenic parameters. This graphic utility, which provides also some basic statistics, monitors the data on periods of 1 hour, 3 hours and 24 hours (see Fig.5). `IS1plot` is extremely useful both to check the system stability (when the instrument is at regime) and to verify that the data slopes are within the safety ranges (during the phases of heating/cooling).

2.3 Cold movements

The GIANO IS2 task must be used to set up each available observational mode. Since there are no motor collision risks inside the GIANO cryostat, there are not specific constraints concerning the sequence of individual setup movements inside the spectrometer. Thus, during the setup, all motor units can be moved simultaneously, with the only exception of the "slit wheel", since slits need to be positioned with an accuracy of $< 1/10$ of their widths (a few microns). To achieve this accuracy, which we required to guarantee instrumental high level performances, we used a wheel driven by a worm gear with large backlash in combination with an IN/OUT mechanism (the "fork"), which extracts the slit-holder from its socket and presses it against three fixed kinematic points. Thus at software level, the movement of the "slit wheel" is forbidden while the "slit fork" is moving (and vice-versa). On the GIANO optical bench the following motors are mounted:

- 5+1 in/out movements:
  - Shift High Resolution (SHR): insert/remove a prism;
  - Low Resolution (LR): insert/remove a mirror;
  - Imaging mode (IMA): insert/remove a mirror (used only for object centering and maintenance);
  - 2 grating motions (Z1 and Z2, for alignment purposes);
  - SLIT fork: insert/remove mechanical lock (coupled with the slit wheel);
- 5 continuous movements:
  - FILTER wheel;
  - SLIT wheel (coupled with the in/out movement for slit lock);

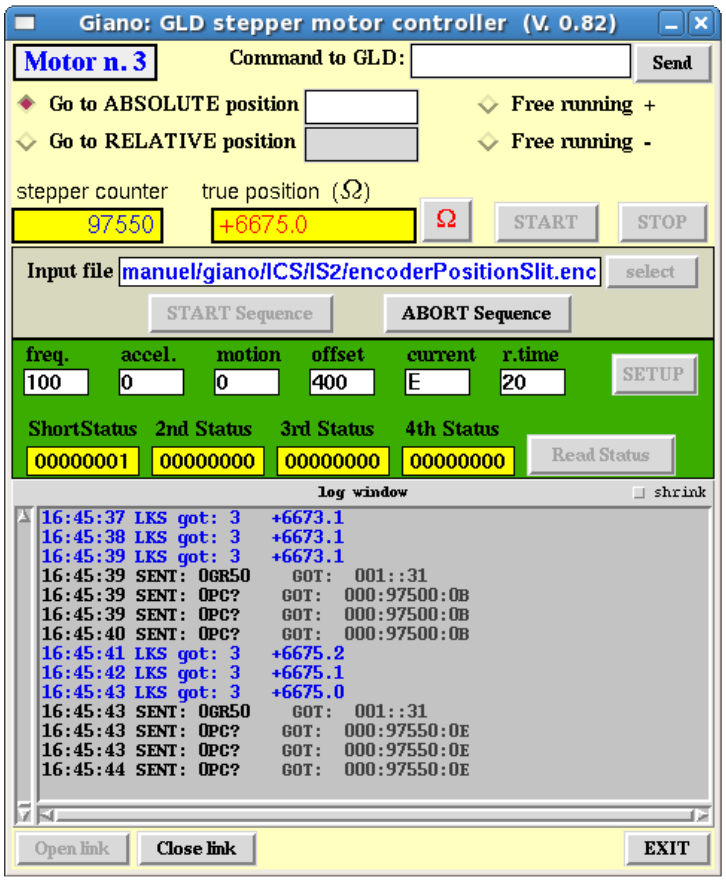

Figure 4. The "gld" low level utility for GIANO motors. Displayed is the execution of a batch sequence.

– 3 array motions (Z1, Z2, Z3: only for alignment purposes).

Each of these 11 cryogenic motors is controlled by a single commercial linear stepper motor drive (*Phytron GLD 20-24*) which is able to keep track of all movements (steps) performed and updates a "step-encoder" value which corresponds to the absolute position of the axis. The GIANO IS2 task make use of a serial communication protocol (*IPCOMM*) to manage each motor. All the 11 *GLD* controllers are serially connected to a *Lantronix* terminal server which manages the remote communications/controls with the GIANO IS2 interface.

To test the GIANO cryogenic motors during the instrumental assembling phase we developed and used a standalone utility (*gld*). This is a Tcl/Tk script which needs to read a configuration file (at start-up level) containing network information (IP and port numbers) for each available (wired) motor. The *gld*, which is simply a "maintenance" tool to be used for all low-level remote controls/operations for each single motor, must be launched as command line adding only the motor number to use (this number must be defined in the configuration file). In Fig.4 we show a typical "testing" session for a GIANO motor.
The GUI has been designed to work as:

- an *entry-command*. The GIANO operator can type and send commands directly to the motor controller without any "filtering" by the interface (with the exclusion of the datagram "checksum" and the "start/end of text" control characters used by *IPCOMM* communication protocol). The log window reports detailed information of what is happening during the execution of the command (whose correctness is under the operator responsibility).

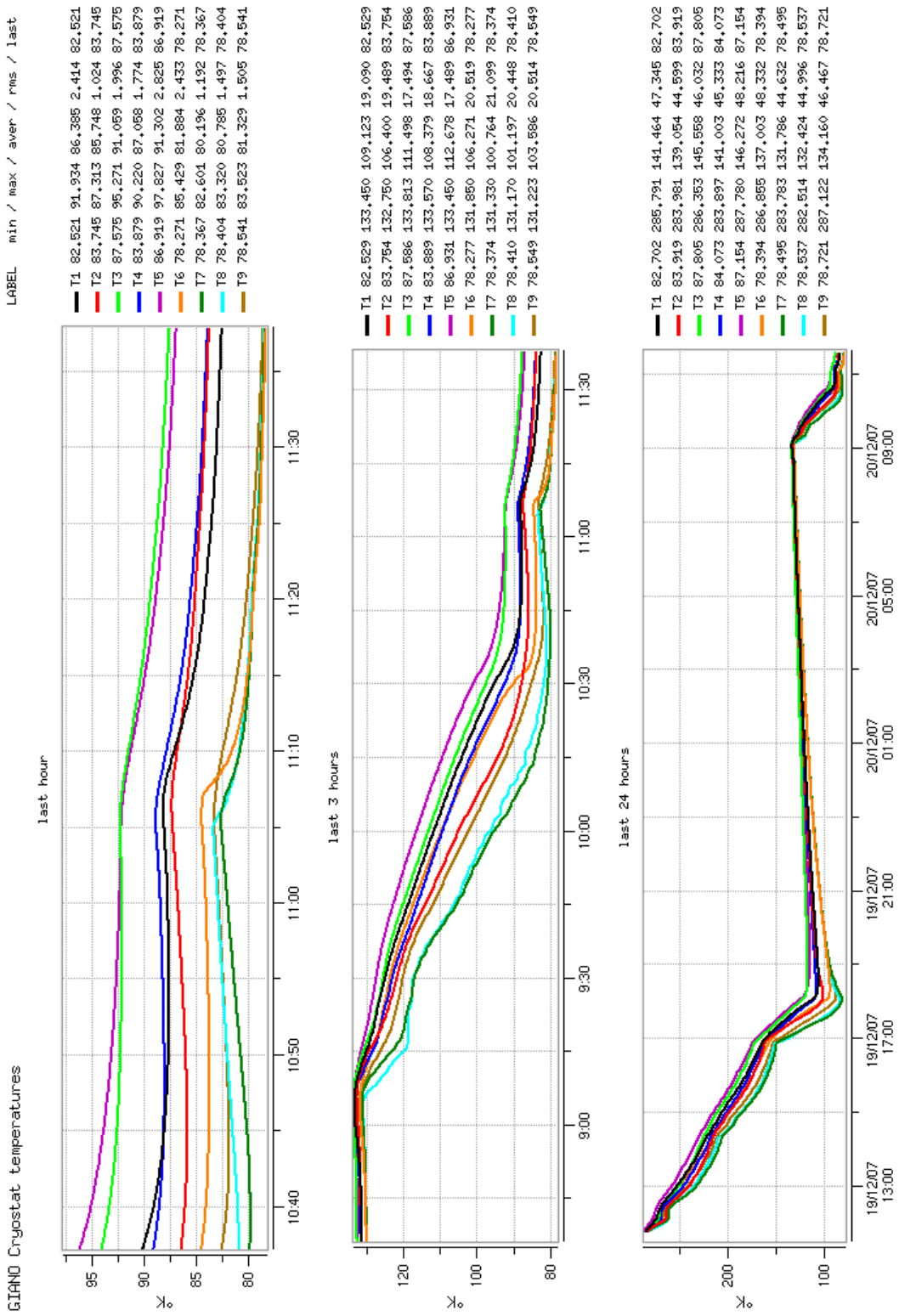

Figure 5. Example of GIANO cryostat temperature monitoring during a cooling session ("strip chart") obtained with "IS1plot".

- a *free* motion manager. An absolute or relative motion (in step units) can be typed on the entry of the *gld* panel: when the motor starts to move the stepper counter updates the actual position (refresh rate of $\sim$ 200 milliseconds) until the motion ends or it is stopped. The motor can be moved also in "free running" mode (useful when a motor is just mounted and must be calibrated).
- a *sequencer* tool. The program allows one to load an ASCII file containing a logical sequence of basics *IPCOMM* commands. Moreover it is possible also to load a "batch" file defining a series of cyclical/repetitive sequences: moving, waiting, reading the stepper counter for a given range of positions and steps.
- a motor *configuration* tool. The "setup" button of *gld* allows to configure and save, for a given motor, a series of *IPCOMM* "parameter commands" like acceleration current, run current, run frequency, type of motion (linear axis or rotating axis) and so on.
- a *status controller* utility. In response to each received command the *Phytron Stepper motor Control unit* transmits back a short status message (8 bit information). Since, at testing level, it is useful to have more detailed information about the motor controller unit, the *gld* can also provide an extended status information request. This latter consists of a datagram response of 32 bits coded, containing the full status information of the *Phytron* device.

For each motor the *gld* utility stores every command sent by the operator and every response received from the controller in a local rotational log file.

3. THE OBSERVATION SOFTWARE

GIANO OS has been devised to coordinate all control activities related to a single exposure which involve several subsystems. These latter are controlled at low level by ICS (see section 2) that directly communicates to OS (for a detailed description see Rossetti et al.[4]). Furthermore, the OS has been planned to be able to communicate with the "external world" (i.e. the other TNG workstations). The communications between the observer and the instrument relate on a keyword based system. To do that the observers can modify some entries in the OS GUI. A super-user mode is also available for GIANO operator who can handle the keywords typing directly a command line input.
The basic actions performed by the OS are:

- sending telescope offset commands to the TNG control workstation, reading telemetry from it and setting the autoguide;
- reading and modifying the position of the DOLORES slide, holding the first pre-slit mirror;
- receiving/transferring images and instrument telemetry parameters to the TNG archive;
- monitoring through ICS the detector status;
- reading from ICS the temperature and pressure values inside the cryostat;
- sending through ICS the commands to the motors, lamps and polarimetric unit;
- reading (when available) the Observation Block sequences from TNG *Flex/SerPiCo*.

To allow standard real time and quick look inspection of the acquired frames we have embedded a SAOImage DS9 (*http://hea-www.harvard.edu/RD/ds9/*) in the main OS GUI. DS9 is a popular and powerful standalone application written in Tcl/Tk which is easily customizable since the *ds9* executable is a *wish* interpreter. This implies that:

- additional custom C code may be added as Tcl extension and loaded at start-up time by using the "package require" or "load" Tcl commands;

- further controls and communications with other Linux tasks can be managed by using the SAOImage *XPA* messaging system;
- additional custom Tcl code can be added at *ds9* start-up time by using the *-source* flag;
- another Tk process can "drive" DS9 by using the powerful "Tk send" mechanism by means of the X server, allowing fully separate Tk applications to control DS9.

We have made GIANO OS drive the embedded DS9 using the methods indicated in latter two items. Furthermore in "our DS9 customization" we have excluded some DS9 menus and buttons and changed the configuration of the geometry for the DS9 display, and the scaling of visualization parameters. To make the loading of an external *fits* file possible, as in the case of incoming images during the observing activity, we have used the "Tk send" mechanism.

To enable a fast evaluation of the data quality we have also implemented in the OS GUI some IRAF (*http://iraf.noao.edu/*) tasks, such as *imexamine* and *qphot*. For practical reasons we have disabled the typing from the keyboard of both single "key" and/or combination "keys" (e.g. the "r" key providing the radial profile in *imexamine*). We have replaced some of these functions with a few "events" triggered by the mouse buttons (i.e.pressing/releasing the pointer buttons).
To avoid slowing up of the system, errors due to mistyping and an incorrect use of GIANO OS which for no reasons is supposed to be a data reducing system, we have enabled only few selected function for each IRAF task.

REFERENCES

[1] Oliva E., Origlia L., Baffa C., Biliotti V., Bruno P., D'Amato F., Del Vecchio C., Falcini G., Gennari S., Ghinassi F., Giani E., Gonzalez M., Leone F., Lolli M., Lodi M., Maiolino R., Mannucci F., Marcucci G., Mochi I., Montegriffo P., Rossetti E., Scuderi S., Sozzi M.
"The GIANO-TNG spectrometer", *Proc. of SPIE* **6269**-46, (2006).

[2] Gennari S., Del Vecchio C., Mochi I., Oliva E., Origlia L., Rossettini F., Tomelleri R., Milani D., Liffredo M., Tommasin D., Roveta G.
"The mechanics and cryogenics of GIANO-TNG", *Proc. of SPIE* **6269**-147, (2006).

[3] Baffa C., Biliotti V., Gennari S., Giani E., Mochi I., Oliva E., Origlia L., Rossetti E., Sozzi M.
"The Versatile Acquisition System of GIANO", *Proc. of SPIE* **6274**-33, (2006).

[4] Rossetti E., Montegriffo P., Baffa C., Giani E., Oliva E., Origlia L.
"GIANO: software design and acquisition facilities", *Proc. of SPIE* **6274**-85, (2006).