PESO - The Python Based Control System of Ondřejov 2m Telescope

Petr Škoda, Jan Fuchs, Jaroslav Honsa
Astronomical Institute, Academy of Sciences of the Czech Republic
251 65 Ondřejov
skoda@sunstel.asu.cas.cz

Abstract

Python has been gaining a good reputation and respectability in many areas of software development. We have chosen Python after getting the new CCD detector for coudé spectrograph of Ondřejov observatory 2m telescope. The VersArray detector from Roper Scientific came only with the close source library PVCAM of low-level camera control functions for Linux, so we had to write the whole astronomical data acquisition system from the scratch and integrate it with the current spectrograph and telescope control systems. The final result of our effort, PESO (Python Exposure System for Ondřejov) is a highly comfortable GUI based environment allowing the observer to change the spectrograph configuration, chose the detector acquisition mode, selecting the exposure parameters and monitoring the exposure progress. All the relevant information from control computers is written into FITS header by the PyFITS module and the acquired CCD frame is immediately displayed in SAO DS9 window using the XPA calls. The GTK based front end design was drawn in the Glade visual development tool, giving the shape and position of all widgets in single XML file, which is used in Python by simple call of PyGlade module. We describe our experience with design and implementation of PESO, stressing the easiness of quick change of GUI together with the capability of separated testing of every module using the Python debugger IPython.

Introduction

At the end of 2003 a new CCD camera system for Ondřejov coudé spectrograph was bought. It was the LN2-cooled VersArray 2048B with EEV 2kx2k thinned chip from Roper Scientific (former Princeton Instruments). Unfortunately the provided data acquisition SW is not suitable for astronomical purposes (targeted rather to laboratory spectroscopy) and, moreover, it is available for MS Windows only. As we needed to incorporate the detector in a complex structure of several Linux based computers, which is called by the control system of another CCD camera as well, we decided the write the whole new data acquisition program from the scratch. After short evaluation of requirements we have decided to use Python. Now we see that this decision was very clever and the system was ready in three month (including the GUI) designed by one programmer (who just started to learn Python) and one astronomer.

Detector control system (DCS)

The DCS (Linux) computer contains the Roper PCI card connected to Roper ST-133A controller by the so-called TAXI cable. The controller is joined with CCD camera head by another shielded cable.

PESO runs on DCS computer. It provides the comfortable GUI interface for commanding all instruments, gathers required information from TCS and SCS that is put into FITS header, creates the FITS file from CCD data acquired, monitors progress of exposition, maintains observing logs and sends the final image to DS9 to display.

It speaks to the CCD controller using binary PVCAM library provided by Roper Scientific. They provide only closed-source shared library PVCAM with header files, simple examples, binary driver for the PCI card and microcode for upload to the controller at the library initialization phase. Unfortunately, the binaries are only for kernel 2.4. Kernel 2.6. is still not supported. The PVCAM library provides only low-level functions for setting the readout parameters (speed, geometry of sub-region, binning, shutter mode), getting the status (temperature, integration mode) and starting the exposure (including opening shutter). The data received are accessed only as a pointer to particular memory structure, no data file creation is done.

The problem of internal timer

The system provides the internal timing on exposure but lacks a function of immediate readout of chip upon request. We have solved the problem by using the provided triggered mode when the shutter opens and closes according to the TTL level on external SYNC connector. We send this signal to the cable from auxiliary TTL IN/OUT connector by low-level PVCAM library function (setting certain bit to 1 to open shutter). All the timing has to be done by PESO and when the readout should occur the command to integrate for 0 sec is sent to the controller - so the readout occurs immediately.

Spectrograph control system (SCS)

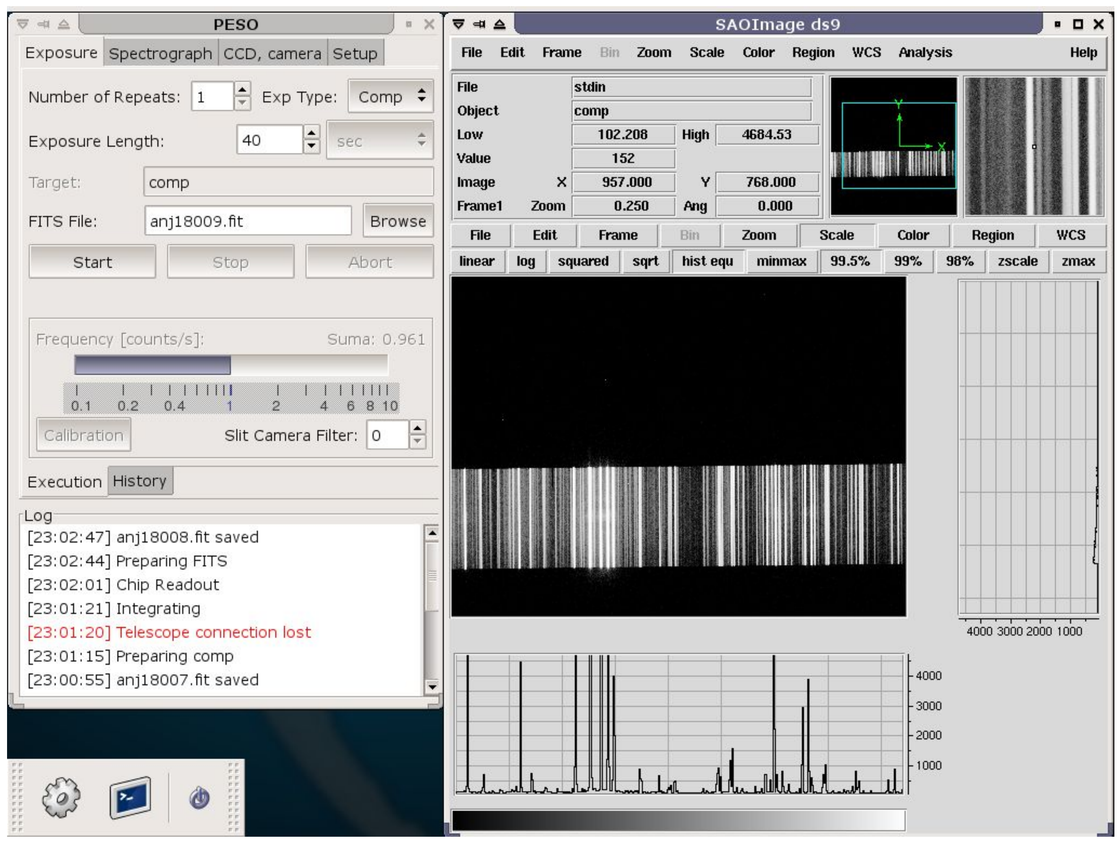
The control system of coudé spectrograph at Ondřejov observatory 2m telescope was already described. It is based on a number of micro controllers driving various units that are on the same serial bus. A simple daemon running on control PC (Linux) sends them commands and reads their status through a dedicated serial card every second. This daemon listens on given UDP socket thus allowing other control computers to activate the spectrograph units by simple sending of two character commands to particular TCP/IP port.

Telescope control system (TCS)

The TCS is based on Real-time UNIX-like system QNX. The information about telescope position, focus, air pressure, temperatures and humidities inside and outside dome is gathered on request through rsh call of TCS and saved in a text file at the beginning of exposure. After some parsing and conversion the relevant information is obtained to be included into the FITS header by DCS.

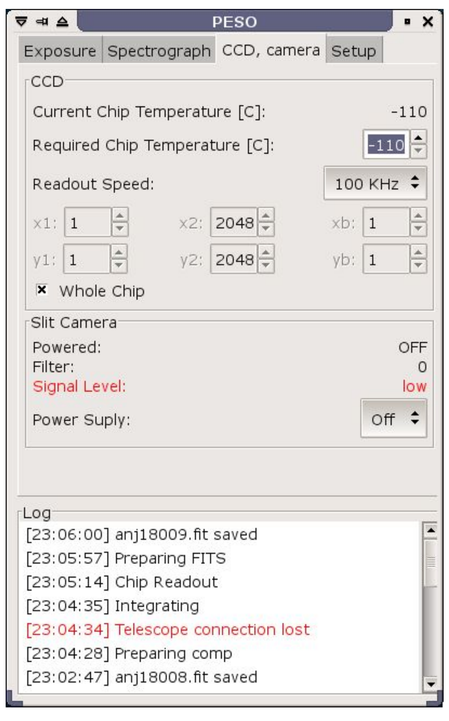
PESO description

The PESO consists of four main tags containing widgets for controlling main areas of CCD acquisition process. The systems requires the observer to do the things in a logical and chronological manner and complains if something important was left out (e.g. observation of a star requires first its name to be given as well as names of observers). On the other hand, PESO works with a number of reasonably predefined values (either built-in or read from configuration files and observing recipes) and can be extended easily. PESO is a Python script that spawns several threads running in parallel. It reconfigures the widgets on-line according to current state of the exposure (e.g. disables certain buttons, creates logs, history etc...). Thus it allows to turn the grating or filters while changing the CCD parameters or allows to display previous exposures while integrating another and at same time it shows the progress bar together with current frequency of photon counts from exposure meter. When the created FITS file (using PyFITS) is ready it is sent for display in DS9 through XPA interface.



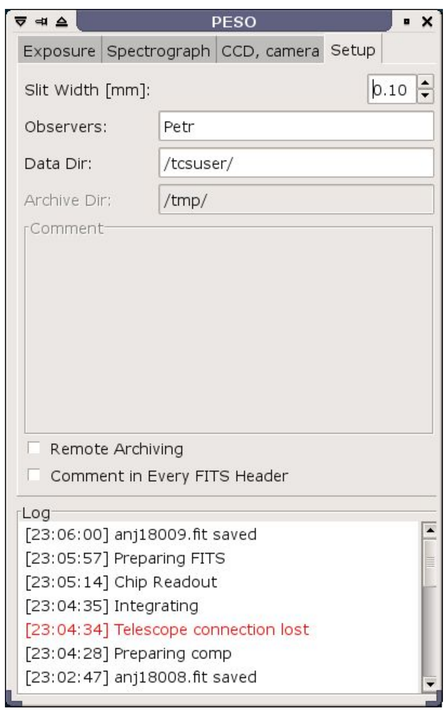
CCD and Camera tab

Here the chip readout parameters (speed, sub-frame size, binning) and chip temperature are set. The slit-viewing camera is powered from here as well. The Roper does not allow to change the gain by SW - only by HW switch.



Setup tab

Before the start of observation - global values are required to set name of observers and slit width (it must be set manually). The path where the FITS will be saved can be changed. If the Remote archiving mark is checked, the copy of the same files is sent to (preconfigured) archiving directory (usually NFS mounted on RAID array). The special comment can be put in every following FITS header (if checked) until changed.



Exposure tab

It is the main window. From here the exposure is started and its progress monitored (sub-tab Execution) or the all previous frames may be displayed together with most important header items (sub-tab History).

The type of exposure (flat, zero, comp, dark, target) is selected here, if it is the target the name of object is requested or a list of all objects exposed in current session is shown. At the same time their last exposure time or count value is preset. If calibration (flat, comp) its recommended exposure time is preset according to the spectrograph setup. Every exposure may be repeated several times.

The stellar exposure may be finished either after elapsing given time or after accumulating given number of counts from exposure meter (option button [sec]/[Mcounts]). The FITS name is auto-incremented according to strict scheme (first two letters represent year and month) but may be changed.

Once a exposure is running, the progress bar shows the elapsed time and on star the count frequency and total sum of counts. The exposure may be prematurately stopped (with immediate readout) or Aborted (without chip readout). The Calibration button recalibrates the bar to 1 if the star gets fainter (cirruses).

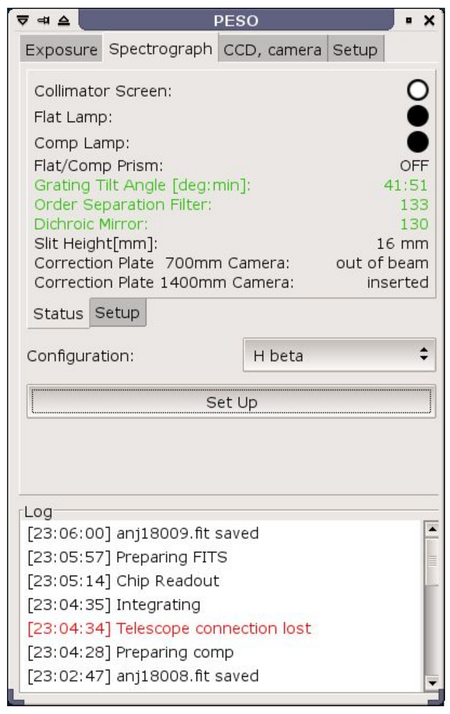
The function of changing the slit camera filter is repeated here for the convenience of observer (only the Exposure window is required if the same configuration is maintained, but the filter has to be changed for different stars).

Every action is logged to logfile and shown in log window - warnings and errors are in red. The warning sound is played as well. The given sound is played at the end of exposure and another if the signal of exposure meter is falling down rapidly - the star goes of the slit.

Every FITS file after its creation is send by XPA interface to DS9 for preview including the x and y slices. Starting of some reduction pipeline is considered in future as well.

Spectrograph tab

here the spectrograph configuration is given - it can be loaded from configuration files (e.g. for H beta) or values from last observation are read. The configuration may prepare the recommended exposure values flats and arcs shown in the Exposure tab. The values can be further modified in Setup sub-tab or watched (even moving - eg. filter wheels) in Status sub-tab



Conclusion

We have proved that Python is well-suited for rapid development of instrument control systems. It allows the quick design of GUI interface using the XML-based Glade development environment, easy calling the external library functions (even close-source) by encapsulating of C calls in Python objects and provides complex process control using its multi-threads module. Its usage for astronomical data acquisition is eased by PyFITS module and its TCP/IP socket module. The debugging of such a complex system is very simple using the IPython thread hooked to the running application.

The GUI development

One of the most time consuming task is the GUI design. We have done it in extremely easy and quick way using the GTK library together with Glade-2 development environment. All widgets are simply prepared by their visual placing and changing their properties and the final GUI is written in single XML file PESO.glade. This file is used by Python *gtk.glade* module to create the objects from all widgets. Thus the GUI is built after PESO startup in memory. The change of widget objects properties is done interactively and the current status of widgets can be detected by the calls of this module as well.

The calling of PVCAM functions

The detector control is done by calling internal functions hidden in PVCAM library from their C wrappers. The C wrapper is C function returning pointer to Python object and accepting parameters in a special structure. Such a library of C functions is then recognized as Python module. All PVCAM functions are finally called as functions of this module objects - even interactively from debugger window.

Timing loops

In PESO there run several threads all the time (including gtk event threads) and some are spawned upon request of particular function (e.g. timer of exposure integration, spectrograph setting control - including checking individual timeouts for every device). To allow this complex behaviour the *threading* module must be used and if some changes in widgets are needed, the *gtk.threads_enter* must be called on entry and *gtk.threads_leave* after leaving this function.

Debugging

The IPython debugger is a module spawning just another thread attached to the terminal window from which a running process maybe be interactively controlled. The functions may be called with custom parameters, the values of variables may be examined, the properties of GUI widgets changed etc. Very comfortable !

Presented at ADASS XIV, IPAC Caltech, Pasadena October 24-27, 2004