Software automation on TELESTO

Louis Dumas
louishenridanielpepito.dumas@epfl.ch,
under the supervision of:
Stephan Hellmich
stephan.hellmich@epfl.ch,

Laboratoire d'astrophysique de l'EPFL
January 2022

Contents

1 Introduction

TELESTO is a F\3.8 600mm reflector telescope, fig 1.1 by Officina Stellare that was installed in 2017 in the AstroDome (AD) at Sauverny Observatory (Geneva -CH). For the last five years, its use has been limited to pedagogical purposes for the students of the University of Geneva (UNIGE) and the Ecole Polytechnique Fédérale de Lausanne (EPFL).

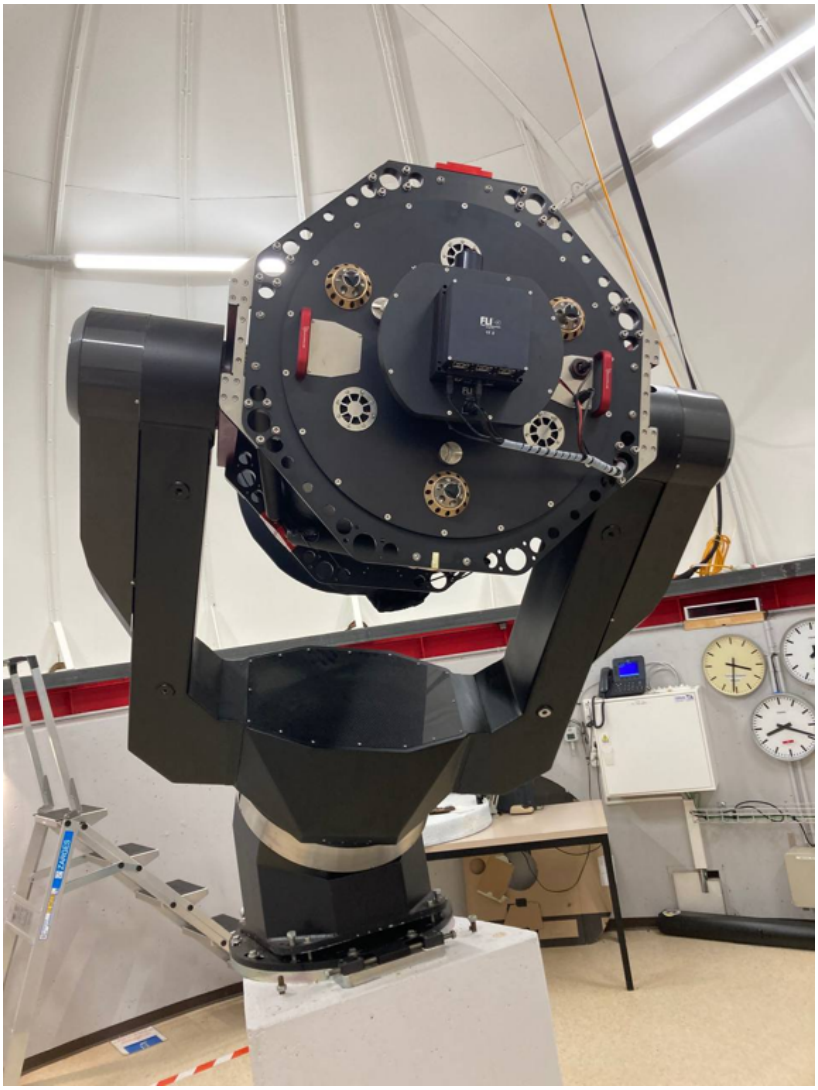

Figure 1.1: Picture of the TELESTO

The need for an automated, easily accessible and not necessarily powerful telescope arose with the recent research domain of satellites, debris and asteroids detection and tracking. Indeed, space congestion, particularly in Low-Earth orbit(LEO), has become a major issue in the last decade. It poses a major threat to astronauts and satellites security, while also deteriorating observations. Moreover, the interest of students of the Space Sustainability Awareness (SSA) association from EPFL for TELESTO now justifies the automation. The goal of the software automation is to avoid interactions as much as possible with all the initial software providing a centralized program in order to use the TELESTO. We will first describe the initial software installation of the TELESTO, how the different softwares interact and problems they could cause, then we will describe which steps in the automation were achieved and how. Finally, we will talk about what kind of problem were encountered and then as a conclusion we will give some indication on what could be done next. Note that the first part will mainly be taken from the prework done during the previous semester by Kent Barbey[1].

2 Initial software configuration

2.1 List of software

The TELESTO is controlled by 3 softwares communicating together. They are all installed on the computer in the control room of the Telescope. The computer's OS is Windows 10.

2.1.1 OSBus Controler

OSBus Controller is the software provided by *Officina Stellare* to control some hardware components of the Telescope. It is used to do dome communication, shutters control, fan control and primary collimation. The main windows is presented with figure 2.1

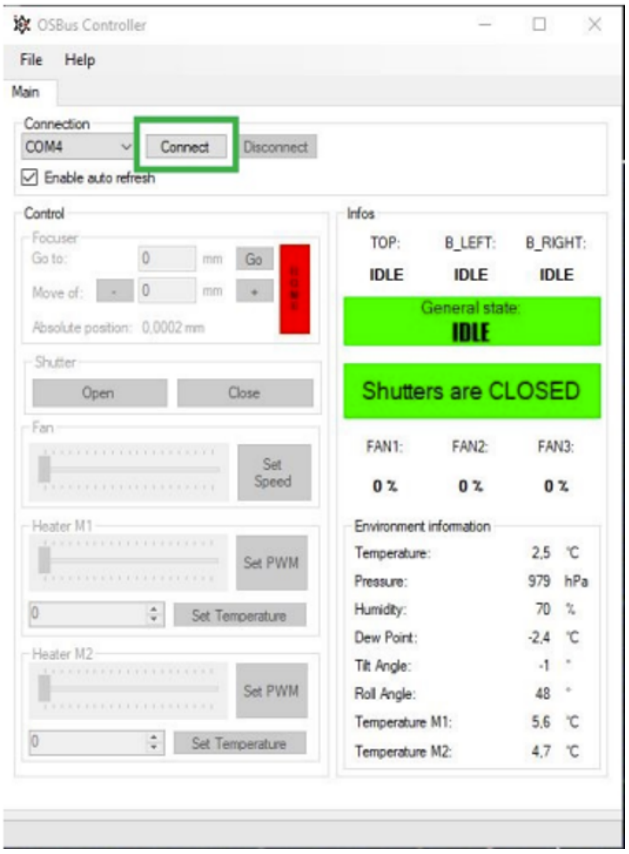

Figure 2.1: OSBus main tab

2.1.2 Maestro

Maestro is an ATCS (Astrometric Telescope Control System) user interface (see figure 2.2). It manages all the astrometric part, the pointing model, the mount, the sideral tracking, the motors speed, the alignment and a lot of other parameters. As it is not user friendly at all we use *TheSkyX Pro Edition* (abbreviated as *SkyX*) to send the correct informations to *Maestro*.

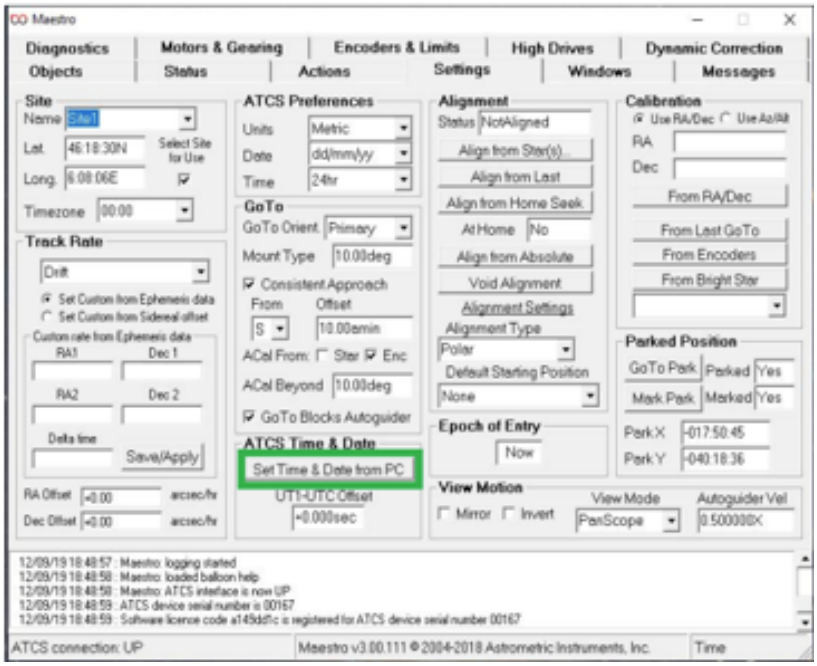

Figure 2.2: Maestro main tab

2.1.3 TheSkyX Pro Edition

Finally, *SkyX* is certainly the most important software from the user view. It is a user friendly interface used for observation. It communicates with *Maestro* through ATCS to perform operations. Anything an user can do for an observation can be done with *SkyX*, like taking pictures , pointing, guiding, seeing object specification among another things. It has an integrated planetarium (2.3) and multiple catalog of celestial object. It makes observation really easy. And the most important thing is that you can perform operation in *SkyX* by sending *JavaScript* string through TCP port as descibed in the user manuel[2].

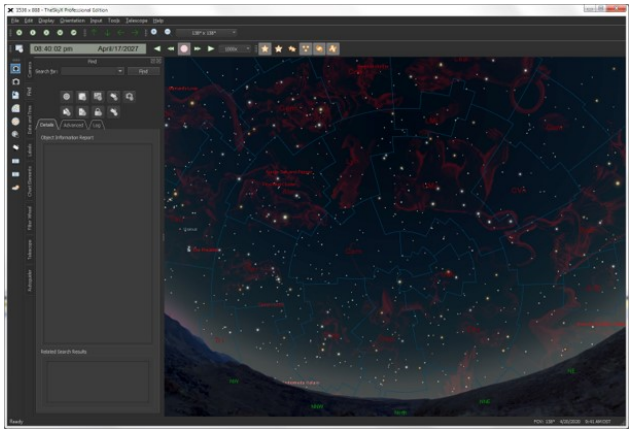

Figure 2.3: *TheSkyX Pro Edition*

All these interactions can be described with the following figure 2.4:

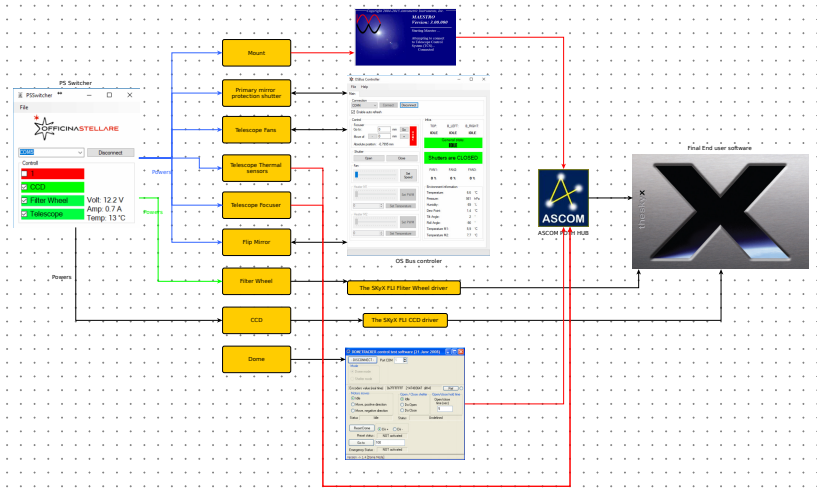

Figure 2.4: *TELESTO softwares interactions*

2.2 Software environment

In addition to the three primary software tools, there are several other software options available. For the purpose of this discussion, we will focus on just two of these additional tools. First there is the plone[3][1] that contain a lot of resources for the TELESTO like documentation for *SkyX* or the starting sequence of the telescope. There is also the remote storage that contains all observation data.

[1]plone is a web content management system. In our case it mainly serves as database for the documentation and pictures

2.3 Issues with the current software setup

There is a certain amount of problems with the current software installation. First, all the interaction between the 3 softwares leads to a complicated starting sequence as described in the corresponding document on the plone[3]. Moreover there is some design error that can break the pointing model during this sequence. It will be describe more precisely in the section 3.4.2. The documentation of the different softwares are either very complex like *SkyX* with a user manual of almost 700 pages, or non-existant like for *OSBus Controler*. The interface of *SkyX* is sometimes very confusing with a lot of settings and parameters that are not clear. And finally there is no way to point or track satellites which is very important in the usage of the TELESTO.

3 Automation: Benefits and Implementations

3.1 Automation priority

As we saw in section 2.3 the main problem of the current software installation is the complex usability. The idea of the automation is to provide a single program that would make the users able to do everything in this program. It must at least automate the starting procedure and take in charge all basic feature of *SkyX* for observation like pointing, taking images, selecting filter and tracking satellites and debris.

3.2 Overview of satellites coordinates computations

The main feature we want to implement is the satellites localisation at a time t. Here the detail of how to compute satellites coordinates. First we need to define the epoch, which has to be as close as possible to the observation time. Then we need some orbital elements to make computation: mean anomaly at epoch M_e, mean motion n, inclination i, eccentricity e and semi-major axis a as described in figure 3.1. With these elements will be able to do a sequence of computation to get coordinates:

- first we compute the mean anomaly M:

$$M = (n(t - epoch) + M_e$$

- then we compute the eccentric anomaly E by solving the following equation with Newton's method:

$$M = E - e\sin(E)$$

- after computing eccentric anomaly we have to compute the true anomaly ν:

$$\nu = 2\arctan(\sqrt{\frac{1+e}{1-e}}\tan(E/2))$$

- finally, we compute semi-major axis a from the following formula:

$$a = \frac{T^2 Gm}{4\pi^2}$$

with T being the period extracted from n, G the gravitational constant and m the mass of the earth.

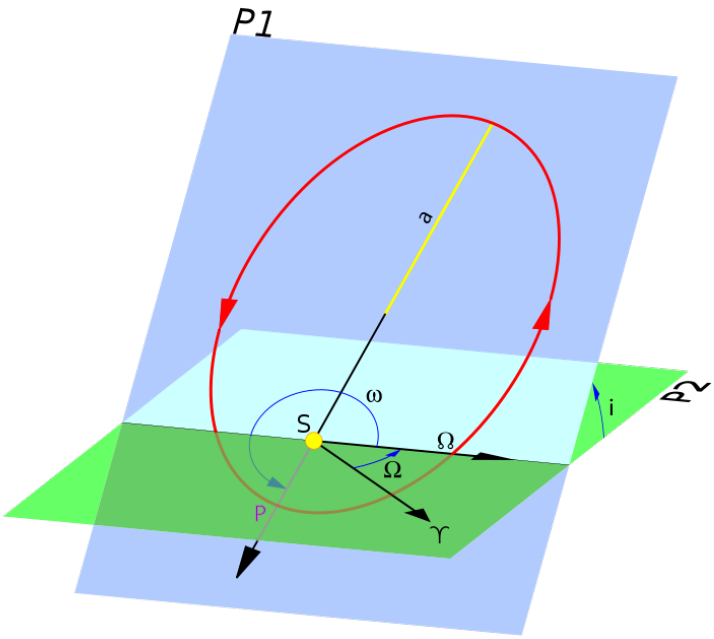

Figure 3.1: Orbital elements: a semi-major axis, e eccentricity, i inclination, Ω right ascension of ascending node, ω perigee, ν true anomaly, E eccentric anomaly, M mean anomaly

- From a and ν we can compute Cartesian coordinates as the following:

$$r = a\frac{1-e^2}{1+e\cos(\nu)}$$

$$x = r\cos(\nu)$$

$$y = r\sin(\nu)$$

- then apply the following rotation to align the previous plan with equatorial plan:

$$\begin{bmatrix} \cos(i) & \text{-}\sin(i) & 0 \\ \sin(i) & \cos(i) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- we need to apply another transformation to get geocentric coordinates.
- Finally, using geocentric coordinates and the observer position on Earth we can calculate topocentric coordinates of the satellite.

Hopefully, we can avoid all these complexes computations in the program using libraries as describe in the next section.

3.3 Implementation

We decided to do a command-line interface called *TelestoInLine* because it is a more convenient way to avoid interface complexity and as we do not need too much commands it will have a light documentation. For the moment our program implements basic pointing, satellites tracking, automated TLE loading, a part of image managing, observer customizable settings and a beginning of starting procedure automation.

3.3.1 Programming language and libraries

Programming language: Python

We decided to use python, despite the use of JavaScript by *SkyX* for multiple reasons. First, python is a way more robust language than JavaScript which is known for a lot of inconsistencies. Secondly there is a lot of scientific oriented libraries on Python that will be useful. Finally, as this is a long term project a high-level language is preferable to make it easy for the next person to add features.

Libraries

We use 4 different libraries:

- the *cmd* library[4] to implement the interface, it is a framework to easily implement line-oriented interpreter. We said earlier that we implemented a command-line interface, it is not exactly right because we cannot run the command we created in any terminal. We must launch *TelestoInLine* to use our commands. It seems to be a constraint but it is a security measure. As we have a very precise starting procedure we don't want the user able to launch any command before the starting procedure is done. So having a specialized environment to run commands is the best way to avoid this kind of problems.

- the *skyfield* library[5] is a astrometric oriented library that made coordinates, ephemeris computation and astronomic information loading very easy, it will be the core of satellites coordinates computation.

- the *PySkyX* library[6] is an interface that made able python to communicate with *SkyX* sending JavaScript strings over TCP port. It has a lot of built-in functions for pointing, imaging and telescope settings in general.

these module interactions could be found at figure3.2

3.3.2 TelestoInLine: description

TelestoInLine is, as described above, a command-line interpreter with a custom set of commands. The current section's purpose is to briefly detail how TelestoInLine works and how to use it. First, the program is made of 9 commands. Their documentation is available in appendix B. To implements the majority of our features we decided to communicate with *SkyX* instead of directly using *ASCOM* driver. This is easier to code and avoids a lot of

hardware and low-level problems. However, we need to find a way to communicate using JavaScript so we use *PySkyX*. To point at stars or planets we simply send instructions to *SkyX*. Indeed it recognises these types of target. But *SkyX* does not have data on satellites or debris coordinates. We then have to compute the coordinates by ourselves. When one wants to point at a satellite, the program will verify that it exists in the different TLEs it has loaded. We then compute the coordinates that the satellite will have one minute later using *Skyfield*. Then the telescope will point to these coordinates and track the satellite by recomputing coordinates until we stop the process or the satellite become to low in the sky. Note that all pointing and tracking functions are not usable if the starting procedure is not launched beforehand. All other command are trivial implementation using *PySkyX* or *Skyfield*.

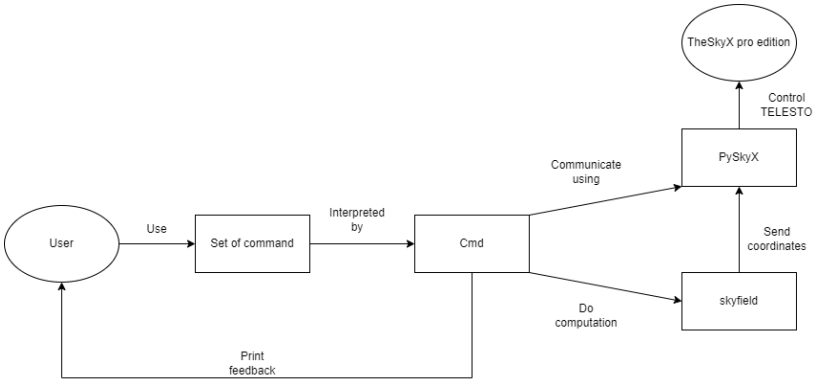

Figure 3.2: TelestoInLine logical scheme

3.4 Problems encountered

We encountered several challenges on the automation, understanding the different APIs, the software configuration and how they interact were the main challenge. Those challenges were solved but there are 2 main problems that either slowed down the project or revealed real infrastructure issues.

3.4.1 Computer problem

The first problem was the programming environment on the computer in the control room. It seems that people who previously tried to write software or automation script had not a clear idea of what was install on the computer. There were multiple versions of python, all being in conflict with each others leading to some python commands not working for program launching (py and python commands). We had to uninstall all python versions an reinstall one version and reconfigure environment variables. It would be a good idea to have a documentation of what is already installed on the computer about the programming framework or even better a virtual environment like *Docker*[1] where we are sure that everything is installed correctly.

[1]Docker is is a platform for launching software in secure container, https://www.docker.com/

3.4.2 Pointing model

The second problem is the biggest one. There is a real lack of understanding of the interaction between all softwares leading to great design fault and a fairly arbitrary startup procedure. Unfortunately we made an error in the starting procedure during testing session. It seems to have erased pointing model offset. The problem was supposed to be fixed by the engineers in charge of the TELESTO but it seems that it hasn't been done yet.

3.5 Testing

We were able to test the telescope movement, image binning and the following procedure. However, We could not test pointing precision for 2 reasons. First during the whole semester the weather was very cloudy and the only observation window was during the 11^{th} week. Secondly, because of problem we describe at subsection 3.4.2 the pointing is malfunctioning leading to pointing error of 1°. In the last news, it wasn't fixed yet. So it is pretty sure that the TELESTO does not point at the right place for the moment.

4 Project prospect and conclusion

4.1 Prospect

There is a lot more work to do before having a full automation of the TELESTO. The list below describe some ideas to implement, as well as incomplete features:

- Finishing imaging implementation. For the moment we can't take pictures from TelestoInLine directly, we still need to use *SkyX*. It would be great to have this possibility and implement it correctly. It includes taking picture, dithering, filter selection and any other feature that seems relevant. However, it implies that the tracking part is done asynchronously to make user able to take a picture while tracking a satellites.

- Automate the starting sequence. For the moment we just load TLE files, initiate observation time and launch all necessary softwares. We do not implement really the starting sequence. A good beginning would be to automate the *Maestro* part. As *OSBus Controller* has no documentation we sent a mail to the constructor to have information. This mail could be found in appendix A.

- Implement a remote usage. From Barbey's report[1] it seems to already have an ssh server on the control computer. For the moment we need someone to turn on the TELESTO, so we need to figure out how make a remote start-up.

- As described in section 3.5, the testing about the precision could not be done because of bad weather and pointing model malfunction. It is an unpleasant but necessary thing to do.

4.2 Conclusion

The software automation of the TELESTO is not complicated by itself. We made great progress by implements some of the core features for the expected usage of this telescope. However, the lack of documentation, design fault in the starting sequence and the necessity to make a lot of roundtrip between Lausanne and Geneva slowed down a lot project progress. We hope this report gave a good idea of what was already done and what needs to be done.

5 Acknowlegment

I would like to thanks B. Chazelas, G. Chaverot and K.Barbey for the answers they gave me and the time they spent on their work time for this project. I also want to thanks K. Sturrock for his library. If there is any need to question him on how his library works he can be contacted on the cloudynights[7] forum by private message.

Bibliography

1. Barbey, K. *Prework on automating the TELESTO* tech. rep. (Laboratoire d'astrophysique de l'EPFL).
2. *TheSkyX Pro User Guide* Software Bisque ().
3. Chazelas, B. *the plone* https://plone.unige.ch/astrodome/telesto.
4. *cmd library* Python Software Foundation. https://docs.python.org/3/library/cmd.html.
5. Rhodes, B. *Skyfield, Elegant Astronomy for Python* https://rhodesmill.org/skyfield/.
6. Sturrock, K. *SkyX Python Library* https://github.com/kenneth-sturrock/PySkyX_ks/blob/master/PySkyX_ks-27SEP2022.zip.
7. *cloudynights forum* https://www.cloudynights.com/.

Appendix

A Mail

R: OSBus Controller API

Luca Bonato <Luca.Bonato@officinastellare.com>

jeu. 17/11/2022 17:15

Boîte de réception

À :Dumas Louis Henri Daniel Pepito <louishenridanielpepito.dumas@epfl.ch>;

Dear Louis,

Understood, thanks. I doublechecked and this indeed has an OSBUS installed, so you should be able to achieve what you had in mind by following the instructions I sent last time:

- *Control the secondary mirror focuser/tip-tilt (if present) using ASCOM drivers that I can send you*
- *Controlling fans and environmental probes through the .exe version of the BUS interface*

The new software can be used as a GUI or as a command line tool.

Calling it from windows prompt with the -h.

Usually the exe file is in:

"C:\Program Files (x86)\Officina Stellare\OSBus Controller\OSBusController.exe"

Anyway, this is the brief of the command line options:

-h Show help

-s[1-0] Open or close shutter

-f[0-100] set the fan speed from 0 to 100%

-m Execute an homing sequence for M2

-ge return environmental data

-gs get shutter status

-gf get fan speed

Of course some might be out of scope for you (e.g. shutter status and control if you don't have shutters on your telescope)

Best,
Luca

Da: Dumas Louis Henri Daniel Pepito <louishenridanielpepito.dumas@epfl.ch>
Inviato: giovedì 17 novembre 2022 14:13

B Program Documentation

TelestoInLine is a command-line interpreter to automate the TELESTO. It is made of 9 documented commands and 1 undocumented command used as an alias to use Ctrl+C. We describe these commands below.

- *add_catalog* [*type*] [*url or path file*]: *type*:
  - sat for satellites
  - deb for debris

  Save the path or the url or the path file for subsequent execution. If start command was already done load the TLE directly.
- *help* [*command name*]: Describe the documentation for the specified command.
- ?: List all the commands
- *start*: Launch all necessary software, load known TLE, initialize observation time and connect camera.
- *target_celestial_body* [*object name*]: point to the specified object if it exists in the *SkyX* database. Can be launch only after the *start* command
- *exit*: Disconnect camera and close all software and the command-line interpreter. Warning: do not close the program in another way.
- *set_bin*: Change the binning of the camera.
- *stop_following*: Prototype for asynchronous execution of $target_satellites$. Not usable for the moment.
- *target_satellites* [*NORAD ID*]: If the NORAD ID exists in one of a saved TLE compute the position in one minute then track the satellites. Stop if the satellites is too low in the sky or if the user use Ctrl+C. Can be launched only after *start* command