The package of control software for automatic astroclimatic monitor

V. Kornilov, O. Voziakova, M. Kornilov, B. Safonov, N. Shatsky

March 15, 2016

Introduction

Automatic site-testing monitor (ASM) of Sternberg Astronomical Institute (SAI) is installed on top of mountain Shatdzhatmaz at 40 m to the southwest of the centre of the 2.5 m telescope tower, according to the project of SAI Caucasus observatory. The monitor was installed and started up in 2007 in order to collect the most important astroclimatic characteristics: statistics of seeing and vertical distribution of the optical turbulence, statistics of the fraction of clear skies, data on the atmospheric extinction, and the most essential meteorological characteristics — the speed and direction of surface wind, diurnal and seasonal variation of ambient temperature and humidity.

The obtained information will be used for a comprehensive characterization of the site where the 2.5 m telescope will be installed, estimation the perspectives of adaptive optics systems implementation and strategical planning of observational tasks to realize the full efficiency of the telescope. In the future, the real-time measurements from ASM will be used to optimize the schedule of observations at the telescope.

1 Network and software architecture of ASM

Automatic site-testing monitor includes the following computing and networking items:

1. Machine `eagle` — full name `eagle.sai.msu.ru`, the local IP 192.168.10.8. It is installed in the communication room of the Solar station. This machine is used as a gateway of the ASM for external access. There are several services running on this machine: an HTTP-server displaying the current state of ASM, archiving of the measurement data coming from other machines of the ASM and web-cameras image storage.

2. Access points 192.168.10.2 and 192.168.10.3 provide network connection to the other machines of the ASM. The first point is located in the building of optical laboratory of the Solar station, the second one — at the ASM tower at mountain summit about 800 m away from the first one.

3. Machine `omicron` (local IP 192.168.10.5) is installed under the enclosure of the monitor in a metal cabinet. Its purposes are collection of meteorological data, management of the power supply, to control the dome enclosure, acquiring images from the outside (yard) and internal (dome) synoptic cameras (SPC900NC Philips), connected to it via the USB interface. RS485 line for exchange of data and commands with the power controller, dome controller and meteosensors controller is connected to the serial port via custom-made COM/RS485 converter. The machine is running permanently, but in the period from 11:57:00 to 12:03:00 local time it's automatically restarted.

4. Machine `druid` (IP 192.168.10.4) is installed in the same cabinet where `omicron` is installed. It is switched on in the beginning of the measurement session and switched off at the end of the session. This machine is connected to the Meade RCX400 telescope, the MASS/DIMM instrument and the telescope finder/guider camera. Its purpose is to control the telescope and data acquisition of the DIMM and MASS channels of the MASS/DIMM instrument. The data exchange with the finder/guide camera (SPC900NC Philips) is performed via the USB port, with the DIMM-channel camera (Prosilica EC650) – via the

IEEE-1394 interface. The MASS channel is connected through LPT/RS485 converter to the parallel port, and the telescope and its controller are connected to the serial port.

The structure of the ASM is represented schematically in Fig. 1, where the left side shows the equipment installed at the Solar station, and the right side — the equipment installed on the ASM tower and in a trailer located near it which houses the power batteries, the meteo-controller and sensors on the mast of wind turbine.

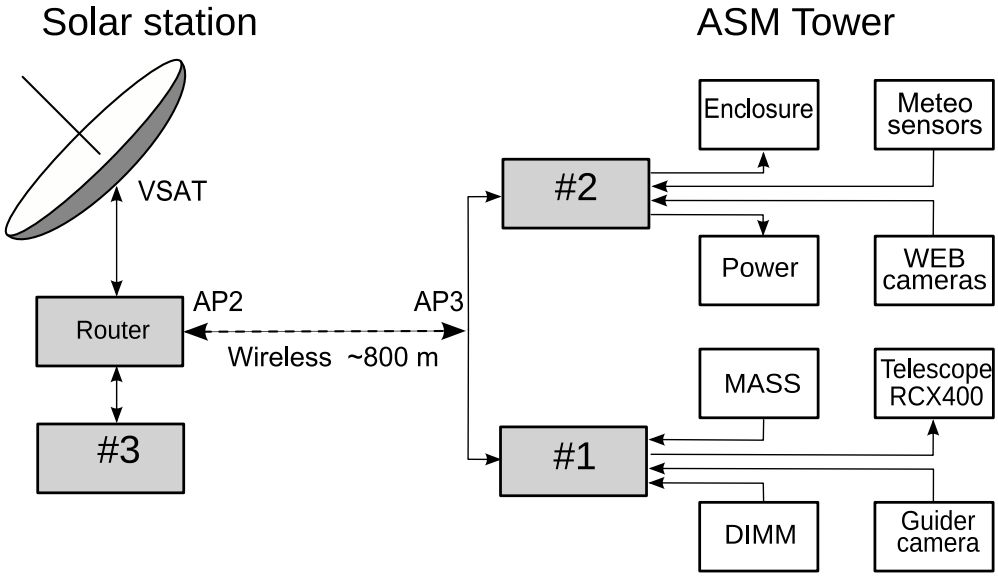

Figure 1: The structure of the automatic site-testing monitor. Here #1 is `druid`, #2 is `omicron` and #3 is `eagle`.

2 Components of ASM software

The operation of the astroclimatic monitor is currently performed by the following software components:

- `tlsp` — the program for control of the RCX400 telescope and finder/guider camera.
- `dimm` — the program for control and data acquisition of the DIMM-channel of the instrument;
- `mass` — the program for control and data acquisition of the MASS-channel of the instrument;
- `shutdown` — the daemon for remote shutdown of `druid` machine;
- `dome` — the program for control of enclosure and power supply for main equipment;
- `monitor` — the program for meteo-data acquisition and power status monitoring;
- `ameba` — the program for scheduling of the observations process.

Data on the sky condition are requested from the program `everest` running on the server of the MASTER project `apollo.sai.msu.ru`, port 15012.

The current state of ASM and other information is displayed with help of components that run on machines `eagle` and `omicron` through the web interface:

1. `meteoAgent` — translation of online data from the `monitor` program for an HTTP-server;
2. `rainAgent` — translation of data from `everest` program for timely enclosure shutting down.

3 Principles for building the software

Our experience of development and operating the MASS/DIMM software obtained in 2002 – 2006 led to changes in the initial approach and requirements to the software for an automatic operation of ASM and other similar automated systems. A detailed discussion is given in the paper "The new software for MASS/DIMM instrument. Control core Turbina-core"[1], where the basic requirements for the software and its elements are represented as well. During the process of designing and programming, we generally adhere to those principles, although some particular ideas have evolved significantly.

Thus, in the process of the development of basic software we tried, where possible, to use the following philosophy:

1. The software consists of several individual software components that run and operate independently of each other, and if necessary one can interact with another. Partition of the software into separate components enables 1) to increase significantly the flexibility by performing some functions at the operating system level, 2) to simplify the process of developing and debugging of the software, and 3) ensure the stability of the whole system, 4) to define and simplify the system logic in a separate software component and 5) to provide a smooth parallel execution of various functions.
2. Each software component implements a single logical task or function of ASM. This implies a division of all the work into separate, fairly isolated and self-contained parts. In this case, priority is given exactly to the problem, not to the maintenance of a device. For example, the program `tlsp` is responsible for pointing the telescope to the desired object, the centering and guiding during the measurement time. For this, the program interacts with the telescope, the special custom-made power controller of the telescope and the finder camera. On the other hand, the enclosure controller is requested from two programs — `dome` and `monitor`.
3. Background operation of the program is a basic feature, so there is no user interface. However, by the default, components start in the foreground. If needed, transition into the background is performed by the program itself but not with help of shell. This approach was selected in order to have the detailed information about the failed startup cases, and then completely release the standard I/O channels. Thus, during normal working all the software components are functioning as "daemons".
4. Reports on progress and error are always recorded in the log file of the program. Some software components do not produce output data (for example, `ameba`). In such cases the log file should contain all the information about the program. The names of the log files are generated using the format `YYMMDD-program.log`, where `YYMMDD` — the program start

[1] `http://curl.sai.msu.ru/mass/download/doc/turbina-core.pdf`

date (calendar date in the evening of the observational night, the date changes at local noon), `program` — program name [2].

5. Controlling of the program is carried out using 1) options on the command line at startup, 2) configuration file data and 3) the commands coming over network connections. The options have the same meaning for all programs because they are handled by a common module and relate to the program context. Configuration files also have similar structure but contain the settings of the functional part of the program and external devices. Commands coming via network connection, initiate a specific action of a separate program. As a rule, these commands do not change tuning parameters.
6. The format of commands of operational control is described in "Supervisor program User Guide SV version 0.24"[3] and was used earlier in the MASS software and other projects. Using text commands of regular format enables efficient manual control, which is important at the stage of debugging the system as a whole.
7. Synchronization of ASM operation is done at the command level. Program `ameba` performs the role of **supervisor** which implements a general algorithm for operation of the ASM. The remaining software components are functioning as servers. There is only one exception: the program `tlsp` uses for automatic guiding the information from DIMM channel — offset of the star image relative to the centre of the working aperture. To do this, it individually sends requests to `dimm` program as a client to perform the corresponding measurement. Since this measurement requires 2 s, the synchronization is carried out according to the principle: the first request is satisfied. The command which came after (and thus rejected with `BUSY` in a response) is repeated by a client within a short period of time if necessary.
8. Logical structure of the software components is built on a common base. The same functions within the program are provided by common modules. In particular, they include the configuration support, error handling, network exchange, analysis of external commands, exchange through the RS485 interface, time and astronomical coordinates. The structure of each program contains a static part and a dynamically generated object which is created in the active (initialized) state — the so-called Device. The static part supports common functions, a device initialization and its destruction.
9. Software components are mainly implemented in C++ with the active use of standard template library. A common programming style is maintained. In a mandatory manner, a source code is stored and developed in the CVS. All software is designed to run on the operating system GNU/Linux. The preferred distribution — OpenSuse 9.2, 10.2, 11.2.

These approaches to development and support of the ASM software correspond to modern ideas of pattern design and have allowed us to develop and debug the entire set of components with minimum expenditure of resources.

4 Location, starting and stopping of software components

Each software component `program` is placed in appropriate directory `/opt/program/`. A structure of these directories is unified. They contain the following subdirectories:

[2] name of the log file of the program `monitor` contains the word `sens`

[3] see http://curl.sai.msu.ru/mass/download/doc/sv_ug.pdf

– `etc/` for configuration files,
– `data/` for all input and output files,
– `data/log/` for log files,
– `data/out/` for output data files.

Executable file is placed in the root directory `/opt/program/`. Also some additional subdirectories can be found there, for example, `/opt/dimm/data/images/` is intended for images obtained on request or as a protocol frames from the CCD-camera of DIMM-channel.

The owner of files and directories is a system user `mass` of `mass` group which has the necessary privileges. In the case of debug or experiments the program can be run by anyone on behalf of `mass`, for this the attribute of the executable file contains an s-bit set.

In the normal mode, these components are run with help of the system utility `cron` either during the computer start (`tlsp`, `dimm` and `mass` at machine `druid`, `monitor` on the machine `omicron`) or at a designated time (`dome` on machine `omicron`, `ameba` on machine `eagle`).

Depending on the situation, launch of the software component can be carried out with various options (keys) on the command line. The following keys are common for all software components:

`-a` — software component and appropriate equipment initialization on startup;
`-b` — transition to background after startup;
`-c name` — use configuration file `name` instead of the default file `/opt/program/etc/program.cfg`;
`-d` — run program in debug mode;
`-p N` — use socket port `N` instead of the default number set in configuration file;
`-i host` — use IP name `host` for machine instead of `0.0.0.0` used as default.

In order to complete the normal operation, the programs are stopped by the command `quit` sent by the program `ameba` to each component. In order to minimize the risk of damage of equipment due to failures of network connections, a special watchdog stops the software after the completion of observations on each machine, normally before sunrise. Its main goal is to turn off the power of the instrument (including high voltage in the MASS-channel) and close the dome.

Manual control of all software components is possible by using network utilities such as `telnet` or `netcat`. When using the `telnet` it is enough to connect to a needed port at the computer where the component works, and to give a text command in accordance with the used protocol. For example, in order to connect to the program `tlsp` using the console of any computer located in the local ASM net one should type `telnet druid 16400`, where 16400 is the program default port number. If the program `tlsp` is running, the connection will be established and any syntactically correct command will be accepted for execution. The utility `netcat` is useful in scripts.

5 Common software modules and commands

As was already mentioned in previous sections, the common logic functions are maintained with help of the common software modules. For various reasons, these modules are used as source files located in directories `infras` and `server` rather than as object libraries. The modules provide 5 basic functions:

– configuring the software component;
– exchange with other programs or user via TCP/IP protocol;
– parsing of external commands and providing their execution;
– exchange with the equipment via RS485 interface;
– handling errors and maintaining the log file.

Structure of configuration files was preserved the same as in the program `Turbina`: a set of separate sections outlined by keywords `Section` and `EndSection` and containing a logically cohesive set of subsections `SubSection` . Each subsection contains specific configurable options related to the particular object of the program. For example, in the configuration file of the program `dimm` section `General` contains two subsections: `Site` and `DIMM`. The first of these subsections contains parameters that determine the geographical location of the device:

```
SubSection "Site"
    DeviceNumber          MD09      ;device serial number
    SiteName               KGO      ;observatory or site name
    Longitude          2 50 40      ;longitude of the site
    Latitude         +43 44 12      ;latitude of the site
EndSubSection
```

Parameter string consists of its name and value. After reading the configuration file, the value is stored in the program as part of a three-dimensional associative array. Not all of the settings contained in the configuration file are need for the program, these parameters are simply being transferred in the output file and then are used by processing software. This approach reflects the principle that the output data must contain all the information necessary for processing and further interpretation of the information.

In a certain degree, the configuration file resembles the logical structure of the program, because the parameters naturally unite together in sections and subsections corresponding to some program objects. The location of some of the parameters is rather arbitrary, for example, the above `DeviceNumber` may be in another place in the file, since it is not used by the program itself nor by the processing software and is given for reference only.

Exchange of commands and data over a network connection in the form of text messages is provided by module `server`. A mark of the end of the command is a terminal symbol `<CR>`. Functioning of the module is weakly associated with the accepted command protocol and focused on the correct service of network connection and disconnection of the clients and handling error situations. In addition to the server side, the module also contains a class for the client side. Recall that the exchange is based strictly on an individual form — the answer to a command or request will be sent only to that client which has sent a request.

The module for command parsing `comms` supports the command protocol. Through the appropriate interface layer it produces the initialization (command `init`) and parking of the program (command `park`), completion of the program (command `quit`), an extraction of relevant data (command `get`), a modification of software parameters or processes (command `set`) and a basic functions of the program by a command `run`. One of the main functions of the module is synchronization of these commands processing.

The command `get` doesn't modify the component status and may always be executed. Other commands will be executed if the program is initiated. Difference between handling of the

command `set` and the command `run` lies in the fact that the first is performed over time much less than the typical time-out of network communications but commands `run` require more time. So, it (as well as the command `init` and `park`) is executed in a separate thread.

Initialization of the program involves the following steps: reading the configuration file, the creation of program objects, opening the output data file, the connection with the equipment and its initialization. After a successful initialization the program goes into the status `READY`.

Parking is done in reverse order: disable hardware, closing exchange and output files. After parking the program is in status `PARKED`. Also, parking is performed before program will quit after receiving the command `quit`.

Commands `set` or `run` cannot be executed if the software component (and related equipment) is in the parked state. In this state, one can only get information about the general state of the program, for which the parsing module provides 3 `get`-commands by itself:

`get status` — returns the component current status: `OK STATUS=READY` or `OK STATUS=BUSY` or `OK STATUS=PARKED`.

`get ident` — returns name of the component and its current version, e.g. `IDENT= ''Turbina-core(D) Vers. 2.26''`.

`get error` — returns last error description in the same form as in the log file.

In further description of commands of specific programs, these commands will be omitted, as well as `init`, `park`, `quit`.

Programs interact with custom equipment with help of batch exchange via the physical interface RS485. In order to provide this interaction we use a set of software modules (along with appropriate system driver developed at SAI). Front-end modules provide all the necessary functions of exchange, including error handling and emulation or exchange for debugging purposes. They also provide synchronization (deadlock) of requests from the different program streams.

Adopted way of handling error states which may occur in operations of software components is based on the mechanism of generation of exceptions in point of error detection and broadcast it to the upper software layers where errors are processed or simply recorded in the log file. Apart from error states, messages on the main phases of the program execution are recorded in the log file. Extended recording in the log file is enabled when you start the program with the option `-d` on the command line. Typically the programs that interact with the equipment via the RS485 interface record all communications and commands. In the `log`-file of the `ameba` all network commands and received replies are recorded.

The structure of a typical message in the log file is illustrated by the following lines from the log file of the `dimm` program

```
2009-04-29 20:38:06.88 (000) Digital Camera (DCAM 1.31) camera is connected
2009-04-29 20:48:31.82 (000) Client 127.0.0.1:2688 attaches
2009-04-29 20:48:38.29 (000) Images have different max brightness > 0.43
2009-04-29 21:34:54.18 (620) No two star images - Centering frame is empty
2009-04-29 21:35:00.09 (000) Client 127.0.0.1:2944 detaches
```

After the UT timestamp, the error code in parentheses is recorded. It was designed for automatic error handling for the client and for filtering of protocol messages (search for identical or similar errors). Code 000 indicates an informational message rather than an error. A textual description of the error consists of two parts, the filling of which is possible on different logical levels of the program for more simple localization of source errors.

6 Software component mass

The program performs the collection and initial processing of data from the MASS-channel of the device and controls this channel. Principles of measurement is similar to those which we used previously in the program Turbina[4]. Data collection and processing is done in the following temporal intervals:

- `exposition` – duration of the elementary flux measurement (microexposure), is given in milliseconds
- `basetime` – interval (in seconds) for which the statistic moments of relative fluctuations of the flux is calculated
- `accumtime` – duration of the whole mode (in seconds) of measurement

All the fluxes written in the output file, are converted to microexposure interval.
The program provides the following modes — the specialized functions of measurement:

- Normal mode – measuring the scintillation of stars over `Normal mode/Accumtime` time. Minimal star flux `Normal mode/MinimalStarFlux` is checked. The mode outputs "m" and "s" lines after `basetime` integration, and "A" lines on the end of accumulation.
- Background measurement – measuring the sky background without any checking during the `Background measurement/Accumtime`. The mode outputs the "m" line and "A" line.
- Flux estimation – any flux measurement over `Tests/FluxEstimationTime` is typically used when checking an object. The mode outputs the "m" line and "A" line.
- Statistic measurement – measuring of the control light source during `Normal mode/Accumtime` for specified control light level. The mode outputs "c" format line after `basetime` integration, and "C" line on the end of accumulation.
- Counting measurement – measuring of the control light source during `Normal mode/Accumtime` for specified high voltage level. The mode outputs "c" format line after `basetime` integration, and "C"line on the end of accumulation.
- Detector test – the measurement of control light source with `Tests/DetectorLight` intensity for the detector checking during `Tests/DetectorTestTime`. The mode outputs the "m" line and "A" line on the end of accumulation. Error is detected if the resulting fluxes don't coincide with specified `Tests/TestCounts`.
- Statistic test – the measurement of artificial scintillation over `Normal mode/Accumtime`. The mode outputs the "m" line and "A" line on the end of accumulation.
- Exchange test – the checking for a failure of data transmission from the device into computer over `Tests/ExchangeTestTime`. The test outputs only final verdict, e.g. `Exchange Test: It's OK`

The sequence of modes can be specified as a scenario — a combination of mode codes and arithmetic operations. Example: F+20*(N+B), where F — mode of the flux estimation, N — normal mode, B — background measurement. In this example, first the flux is estimated, then pair of normal and background measurement modes is repeated twenty times.

The list of implemented commands:

[4] http://curl.sai.msu.ru/mass/download/doc/main_document.pdf

– **run** or **run normal** – start the normal mode,
– **run background** – start the background measurement,
– **run flux** – start the flux estimation,
– **run dtest** – start the detector test,
– **run stest** – start the statistic test,
– **run etest** – start the exchange test,
– **run stat=ligh** – start the statistic measurement set before control ligh level (0.0–1.0),
– **run counting=hv** – start the counting measurement set before hogh voltage value in Volts,
– **run normal background dtest** – start the few modes in one sequence,
– **run scenario=2*N+F** – start the scenario 2*N+F,
– **run scenario** – start the scenario which was set before,
– **get flux** – request for the flux values, the values for 4 apertures `FluxA=... FluxB=... FluxC=... FluxD=...` are returned, averaged over last completed mode,
– **get data** – request for the measurement results for the completed mode (output as in the file format),
– **get temperature** – request for the inner temperature of the device,
– **get hv** – request for the high voltage status. `ON` or `OFF` are returned,
– **get illum** – request for the field aperture illumination status. The response is `ON` or `OFF`,
– **get light** – request for the control light status. The response is `ON` or `OFF`,
– **get mirror** – request for the viewer mirror position. Response `ON` — the mirror is at optical axis, `OFF` — the mirror is out,
– **set scenario=2*N+F** – set of scenario 2*N+F (for example),
– **set object="..."** – write the information on object in the output file. This is used for data processing and is provided as (e.g.) `20 set object=''7949 Eps_Cyg 20 46 13 +33 58 13 2.46 1.03 K03''`, giving the HR-catalogue number, star name, ICRS coordinates, V− and B−V magnitudes and spectral code for weight function selection,
– **set illum=on/off** – turn on/off the field aperture illumination,
– **set hv=on/off** – turn on/off high voltage for PMTs,
– **set cnt=on/off** – turn on/off record of raw counts over `exposition` into `*.cnt`-file
– **stop** – stop of scenario after currently executed mode,
– **stop now** – abort the mode after current `basetime`.

The results of `basetime`-measurements are written to the output file `YYMMDD-mass.stm` in lines with the prefix "m". Average over `accumtime` results are written as lines with prefix "A". Those lines format is identical to Turbina "m" lines format and contains the following space-separated fields:

1. Line prefix "m" or "A"
2. UT date of the measurement finish in YYYY-MM-DD format
3. UT time of the measurement finish in hh:mm:ss format

4. Average flux in A aperture
5. Average flux in B aperture
6. Average flux in C aperture
7. Average flux in D aperture
8. Normalized variance of flux in A aperture
9. Normalized variance of flux in B aperture
10. Normalized variance of flux in C aperture
11. Normalized variance of flux in D aperture
12. Normalized covariance of fluxes in A and B apertures
13. Normalized covariance of fluxes in A and C apertures
14. Normalized covariance of fluxes in A and D apertures
15. Normalized covariance of fluxes in B and C apertures
16. Normalized covariance of fluxes in B and D apertures
17. Normalized covariance of fluxes in C and D apertures
18. Normalized temporal (lag 1) covariance of flux in A aperture
19. Normalized temporal (lag 1) covariance of flux in B aperture
20. Normalized temporal (lag 1) covariance of flux in C aperture
21. Normalized temporal (lag 1) covariance of flux in D aperture
22. Normalized mixed (lag 1) covariance of fluxes in A and B apertures
23. Normalized mixed (lag 1) covariance of fluxes in A and C apertures
24. Normalized mixed (lag 1) covariance of fluxes in A and D apertures
25. Normalized mixed (lag 1) covariance of fluxes in B and C apertures
26. Normalized mixed (lag 1) covariance of fluxes in B and D apertures
27. Normalized mixed (lag 1) covariance of fluxes in C and D apertures
28. Normalized temporal (lag 2) covariance of flux in A aperture
29. Normalized temporal (lag 2) covariance of flux in B aperture
30. Normalized temporal (lag 2) covariance of flux in C aperture
31. Normalized temporal (lag 2) covariance of flux in D aperture

The example of an "m" line:

```
m 2009-06-29 18:07:27  195.1   332.6    1476    1636  0.30575 0.22430 >>>
0.13019 0.08321 0.25292 0.16987 0.08573 0.15699 0.08648 0.08862 0.20257 >>>
0.15963 0.10496 0.07140 0.17305 0.13209 0.08179 0.12354 0.07904 0.07805 >>>
0.30545 0.22407 0.13013 0.08320
```

For `Statistic measurement` and `Counting measurement`, the results of `basetime`-measurements are written to the output file `YYMMDD-mass.stm` as lines with the prefix "c". Averaged over the `accumtime` results are written as lines with prefix "C". Those lines format is identical and contains the following space-separated fields:

1. Line prefix "c" or "C"

2. UT date of the measurement finish in YYYY-MM-DD format
3. UT time of the measurement finish in hh:mm:ss format
4. Average flux in A aperture
5. Average flux in B aperture
6. Average flux in C aperture
7. Average flux in D aperture
8. Normalized variance of flux in A aperture
9. Normalized variance of flux in B aperture
10. Normalized variance of flux in C aperture
11. Normalized variance of flux in D aperture
12. Poisson parameter of flux in A aperture
13. Poisson parameter of flux in B aperture
14. Poisson parameter of flux in C aperture
15. Poisson parameter of flux in D aperture

The example of an "c" line:

```
c 2014-11-27 20:36:56  0.130   1.425   1.033   0.377  8.77569  0.81917  >>>
1.27365  3.08925   1.1371   1.1676   1.3157   1.1647
```

7 Software component dimm

Modes, formats of the output files and commands of the program `dimm` are described in detail in a special document "Turbina-core(D): Dimm User Guide"[5].

Just as in the program `mass`, the following temporal intervals are used:

- `exposition` – frame exposure of the CCD-camera (in milliseconds),
- `basetime` – interval (in seconds) over which the calculated parameters and characteristics of differential motion of images of star are computed,
- `accumtime` – duration of measurement mode (in seconds).

The list of commands is shown below with a brief explanation of the function related.

- **run** or **run normal** – start the main mode to measure differential motion inside the working window at the CCD frame,
- **run center** – start the measurement of the images positions in whole field aperture
- **run test** – start the measurement of the illuminated field aperture for device parameters evaluation,
- **run scenario=2*N+C** – start the scenario 2*N+C (for example),
- **run raw** – start the main mode with raw data output,
- **run estimation** – start the main mode with duration of one `basetime`,
- **run pictures** – start the record of the working window images into output file,

[5] `http://curl/mass/download/doc/dimm_soft_description.pdf`

– **get offset** – request for the position of common pair centre regarding centre of the field aperture (in pixels),
– **get separation** – request for the separation between images for x and y (in pixels).
– **get flux** – request for the total and maximal intensities for left and right images (counts per exposure),
– **get data** – request for the data. The data of last completed mode is returned in the form of output file string,
– **get mode** – request for the name of current or completed mode,
– **set scenario=2*N+C** – set of scenario 2*N+C for further action,
– **set object="..."** – write the information on object in output file like: `20 set object=''6396 Zet_Dra 17 08 47 +65 42 53 3.17 -0.12 B55''` (see mass component section),
– **stop** – stop of scenario after current mode,
– **stop now** – break the mode after current `basetime`.

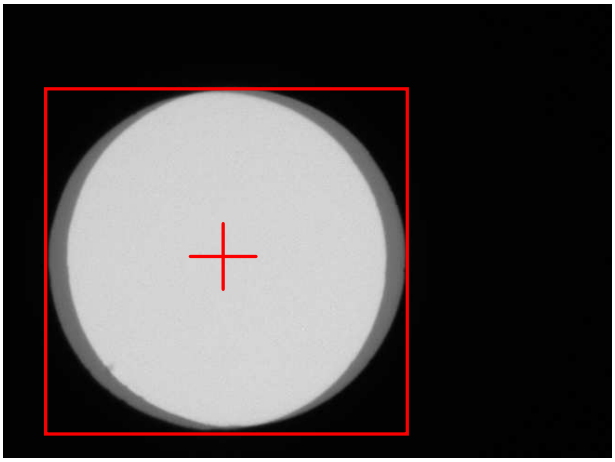

Figure 2: Field aperture view at the frame obtained with the command **run test**

The results of `basetime`-measurements are recorded to the output file `YYMMDD-dimm.stm` as lines with the prefix "d". Averaged over the `accumtime` results are written as lines with prefix "D". The string format is identical and contains the following fields:

1. Line prefix.
2. UT date of the measurement finishing in YYYY-MM-DD format.
3. UT time of the measurement finishing in hh:mm:ss format.
4. Number of processed CCD-frames.
5. Integral flux of the left image, ADU.
6. Integral flux of the right image, ADU.
7. RMS value of the relative flux fluctuations in the left image.
8. RMS value of the relative flux fluctuations in the right image.
9. Maximal intensity of the left image, ADU.

10. Maximal intensity of the right image, ADU.
11. Mean separation between images along x axis, pixels.
12. Mean separation between images along y axis, pixels.
13. RMS value of the separation along x axis, pixels.
14. RMS value of the separation along y axis, pixels.
15. Covariance (lag 1) of the separation along x axis, pixel^2.
16. Covariance (lag 1) of the separation along y axis, pixel^2.
17. RMS noise estimation in x separation, pixels.
18. RMS noise estimation in y separation, pixels.
19. Mean position of common centre of images in x, pixels.
20. Mean position of common centre of images in y, pixels.
21. RMS value of position of common centre of images in x, pixels.
22. RMS value of position of common centre of images in y, pixels.
23. Mean FWHM of the left image, pixels.
24. Mean ellipticity of the left image.
25. Mean FWHM of the right image, pixels.
26. Mean ellipticity of the right image.
27. Averaged background, ADU.
28. RMS background fluctuation, ADU.

The example of "d" line:

```
d 2006-11-19 02:33:59 294 18068 18568 0.232 0.240 1932 2001 42.89 -2.23 >>>
1.286 0.840 1.501 0.594 0.020 0.019 -17.4 16.3 0.62 0.58 3.33 >>>
-0.02 3.34 0.08 15.40 5.47
```

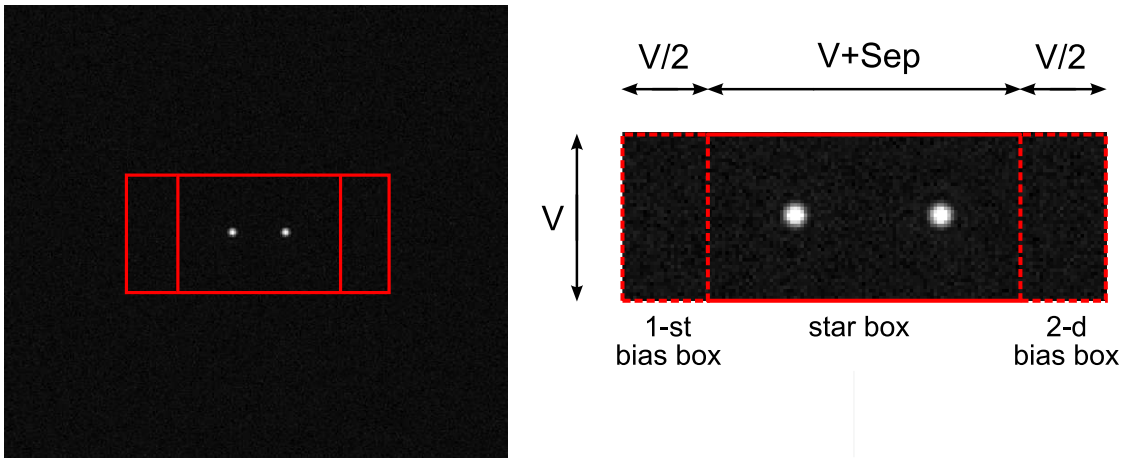

Figure 3: Left: The star image in the frame after command **run center**. Right: The structure of the working window used in the normal mode measurements

8 Software component tlsp

The program `tlsp` is designed to control the 30-cm telescope Meade RCX400, connected to the machine `druid` via serial port (device `/dev/ttyS0`). In addition, the program provides search, centering and guiding the target star from the images received from the finder camera. When guiding is provided by the DIMM data, the program works as a TCP/IP client, periodically requesting `dimm` for the displacement of the star image with respect to the optical axis of the instrument. Such a derogation from the basic principle — the software components interact through a single centre, serving as the supervisor (program `ameba`) — was allowed to encapsulate the functions of guiding inside a single program.

List of commands of the program `tlsp`:

- **run ra**[=...] or **run dec**[=...] or **run ra**[=...] **dec**[=...] – start of pointing the telescope in the equatorial coordinate system. If the arguments are missing, the values set before by the command **set** are used. Arguments are fed as a string in a usual sexagesimal format (see examples below). The pointing includes the turning on tracking, search for the brightest star in the neighborhood of a target (approximately $3^\circ \times 2^\circ$) and a coarse centering,
- **run az**[=...] or **run alt**[=...] or **run az**=... **alt**=... – pointing the telescope in the horizontal coordinate system (see example below). In contrast to the pointing in the equatorial coordinate system, the search for the stars is not made, and tracking turns off,
- **run point** – start of pointing the telescope in the equatorial coordinate system to the target set before by a **set** command,
- **run center** – start the exact star centering procedure in the finder field of view,
- **run search** – start the target star search in a 3×3 pattern of the finder fields of view (which are approximately $3^\circ \times 2^\circ$) and coarse centering (if found).
- **run dx**=... **dy**=... – shift the telescope in horizontal coordinate system in arc seconds,
- **run focus**=*shift* – the telescope focusing, value `shift` is relative shift from current position in some arbitrary units,
- **stop** – stop of the pointing or any other command **run**.
- **set guide=on** or **set guide=dimm** – start the telescope guiding with help of finder camera or DIMM information,
- **set guide=off** – turn off the guiding,
- **set sync=on** – reduce the telescope coordinate system into compliance with the coordinates of the last target. This command is useful to refine the coordinate system in case when centering has required a significant offset of the telescope,
- **get ra dec** – request for the current telescope position in the equatorial coordinate system,
- **get az alt** – request for the current telescope position in the horizontal coordinate system,
- **get guide** – request for the current autoguider status. Possible responses are: `ON` or `OFF`.
- **get scope** – request for the current telescope state. Possible responses are: `PARK`, `SLEW`, `TRACKING`, `STAND`, `SEARCHING` or `CENTERING`.
- **get offset** – request for the position of target star relative to optical axis of the finder (in the horizontal coordinate system in arcseconds).

Coordinates of the pointing target are entered as a sexagesimal string. For example: `20 run ra=''20 46 13'' dec=''+33 58 13''` or `20 run az=''123 23 07'' alt=''+55 44 12''`. Tracking starts automatically after pointing in the equatorial coordinate system and terminates after pointing in the horizontal system or during parking.

During initializing the telescope power is turned on and the UT time, date and time zone are set. It is assumed that the telescope is in position with coordinates `az =''000 00 00''` and `alt =''00 00 00''` (parking position). If the telescope is set in a different position, then its coordinate system turns out to be wrong. During parking the telescope is set in a parking position and powered off.

If the guiding is performed with help of DIMM channel data, then at the same time the position of the stars in the finder camera is measured. The difference between the optical axes of the finder and the instrument is calculated to automatically adjust this shift during work. In order for get the optical axes well matched during the next initialization, the manual update of the parameter `Operations/Centering/OpticalCenter` in the configuration file `tlsp.cfg` is needed.

9 Software component dome

The program is developed to control the ASM dome, power supply for the machine `druid` and the MASS/DIMM device and lighting under dome space. Commands are listed bellow:

- **run dome=open** and **run dome=close** – opening and closing of ASM enclosure.
- **set limit=...** – set the motion limit for enclosure opening. The dome which is fully opened has the position ≈ 700, and closed dome — 0.
- **set power=on** and **set power=off** – turn on/off of the machine `druid` and the MASS/DIMM instrument.
- **set light=on** and **set light=off** – switch on/off bright LED for lighting under dome space.
- **get dome** – request for the enclosure status. Possible response is: `OPENED`, `CLOSED` or `UNDEFINED`.
- **get position** – request for the current position of the enclosure blind in arbitrary units (see command **set limit=...**)
- **get power** – request for the powering of the equipment. Possible response is: `ON` or `OFF`.
- **get light** – request for the lighting. Possible response is: `ON` or `OFF`.

10 Software component monitor

`monitor` is a continuously running daemon to collect information about ambient conditions from three temperature sensors, humidity sensor and anemometer. Additionally it monitors the voltage of the battery and in the cabinet under the dome. Also the program registers the state of two limit switches, one of which is installed on the front door of the car and the second one is reserved. Originally the speed of wind turbine was being recorded, but this sensor quickly broke down and was no longer used.

This program allows only to request for the current data on environmental conditions so no active functions exist.

– **get meteodata** – request for basic meteorological parameters. Returns the values of the ambient temperature `TEMP_EX` (measured on the mast), temperature in the car `TEMP_IN`, wind speed `WIND` in m/s, wind direction `WIND_DIR`, humidity `RH` and calculated dew point temperature `DEW_P` averaged over 1 minute and the maximal wind speed `WIND_MAX` registered in that minute.

– **get support** – request for additional parameters related to the life-support system of the ASM. Rotation rate of the wind generator `GEN_RPS`, battery voltage `V_ACC`, temperature in the box of actuators controller `TEMP_BOX` and voltage there `V_BOX`, state of switches `SW1` and `SW2` are returned.

– **get sky** – request for the data from the clouds sensor. Ambient temperature `TEMP_AMB`, sensor temperature `TEMP_SEN`, sky temperature `TEMP_SKY` with respect to air temperature and sky status `SKY` are returned. Sky status may be 1 – clear sky, 2 – light clouds, 3 – clouds, 4 – rain or snow.

– **get wind** – request for the wind parameters. Mean wind speed `WIND`, maximal wind speed `WIND_MAX` and wind direction `WIND_DIR` are returned. Wind direction is measured from north to east.

– **get power** – request for the power supply parameters. Battery voltage `V_ACC` and ASM box voltage `V_BOX` are returned.

The program logs the measurement data in the output file every 2 s as well as averaged over 1 minute values. Example for the averaged line:

```
M 2008-12-16 09:03:37   -2.3   8.7   8.1  10.4  240  30.8   0.0 -17.2 >>>
  0.0  28.38  14.2  28.00  on  on  12.3  14.2 -20.4 1
```

Field values are as follows:

1. M-prefix.
2. UT date of the measurement finishing in YYYY-MM-DD format.
3. UT time of the measurement finish in hh:mm:ss format.
4. External temperature (the mast) in °C.
5. Internal temperature (the car) in °C.
6. Mean wind speed in meters per second.
7. Maximal wind speed.
8. Azimuth direction of the wind in degrees (measured from north to east).
9. Humidity in percent.
10. A field reserved for the atmospheric pressure.
11. Dew point temperature in °C.
12. Rotation rate of the wind generator (no longer in use).
13. Battery voltage (V).
14. Temperature in the ASM box in °C.
15. Voltage in ASM box (on the actuator controller) (V).
16. Switch 1 door car status (`on` — the door is closed).
17. Switch 2 status (reserved).

18. Ambient temperature from sky sensor.
19. Temperature of the IR sensor.
20. Sky temperature relative to the ambient air.
21. Status of the sky.

Line for current values (prefix m) does not contain fields `WIND_MAX` and `DEW_P`, because these values are calculated at the stage of averaging of the measured data.

11 The software organization for automatic operation

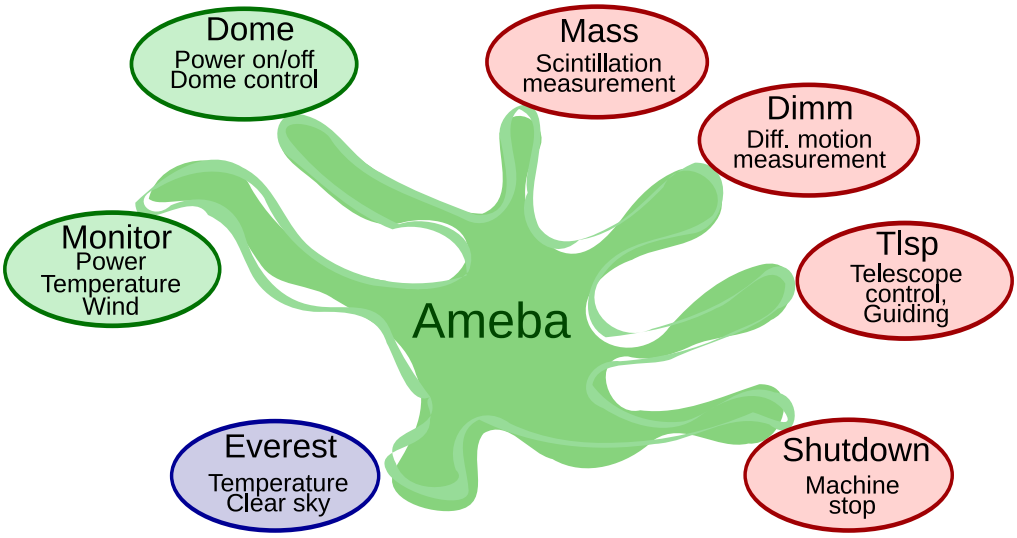

Figure 4: Software structure of ASM and component interactions when operating in automatic mode

General organization of the ASM functioning is provided by the program `ameba` working on the machine `eagle` which is running permanently. In terms of inter-program exchange, it serves as a client for other software components. To simplify the logical structure the program runs in one thread and almost all operations are carried out consistently although this leads to some loss of efficiency. This supervising program is also written in C++ and the system logic is hardcoded there while its parameters are tuned in the configuration file.

The program starts every afternoon with help of `cron` daemon between 17 and 19 hours local time, depending on the season. Somewhat earlier at the machine `omicron` the program `dome` starts to provide the readiness of the equipment to the moment of beginning of observation. At the start, `ameba` computes the moments of the beginning and end of the night under the given conditions: the height of the Sun is $-9°$ and before the beginning of night is in standby mode, polling periodically the programs `monitor` and `everest`.

To start the measurements of the atmospheric turbulence the following conditions must be met:

1. Night is begun under corresponding criteria.
2. Data from the sensor of the sky show that the sky is clear.

3. Wind speed does not exceed the limit.
4. Battery voltage is above the specified threshold.

Result of measurements of relative humidity was planned to be included in the list of the conditions, but it appeared that our particular sensor begins to show strongly underestimated values in high humidity due to fogging.

If the conditions have been satisfied, then session of measurements begins: the set of procedures for the preparation, execution and completion of measurements of the atmospheric turbulence. Each session includes the following steps:

1. Opening the dome, turning on the machine `druid` (the program `mass`, `dimm` and `tlsp` start automatically), the initialization of the software.
2. Select the object according brightness, minimum and maximum altitude above the horizon, distance from the moon, duration of available measurements.
3. Pointing telescope to the object, search the object, its alignment, its verification with the expected flux, start of the guiding.
4. Turbulence measurements over 1.5 hours or until one of the conditions of choice is violated.
5. Background measurement before and after the measurements selected stars, but at least once per hour.
6. In case of failure (object not found, measured brightness is not as expected, a large background) — re-pointing to the following program star.
7. If an error occurred during the measurements (the flux from the object decreased significantly, malfunction of equipment) — re-pointing to the following program star.
8. When a fatal error condition (error 3 times in a row) or a violation of the conditions of observation the session is completed: park components and equipment, close of the dome, turn off the machine `druid`.

After the session termination due to worsening external conditions `ameba` waits for the restoration of conditions for observation. To prevent random effects, the considerable hysteresis has been included in the detection system. For example, if the session is completed because of wind speed is greater than 9 m/s work resumes only if within ≈30 min. wind speed will not exceed 6 m/s. Similarly, if the temperature of the sky T_{sky} exceeded $-21°$C, then the observations are completed and will be resumed, provided that within 20 – 30 minutes $T_{sky} < -22°$C.

After the termination due to error situation (error or measurement guiding, the lack of a suitable object for the measurement, large background, etc.) ASM pauses for 20 – 30 min as well.

Final completion of observations occurs in the calculated end of the night. The program consistently completes the current session of observations, disconnects from the meteo-services `monitor` and `everest` and quits. If the ASM is in standby mode, then the completion happens somewhat earlier, in 20 – 30 minutes before the end of the night because starting a new session is meaningless.

Exact values mentioned pauses and limits for determining the conditions are contained in the configuration file and are refined with gaining experience. Fig. 5 shows the logical scheme of the program `ameba`, on the left: the conditions of the observations, on the right: the process of measurement.

In January 2009, a procedure for photometric observations was added in the program to accurately determine the atmospheric extinction in the sensitivity bands of the MASS and DIMM

channels. To do this, before pointing to the next target of basic program ASM additionally measures flux from 2 stars for photometric binding and one for estimation of extinction. These star are selected from a special list.

For measurements of turbulence the updated list `mass_star.lst`[6] are used. It contains 119 bright stars. From the same list the program selects objects for photometric measurements according to following conditions: the star should not be a variable, must have a color index $B - V$ not greater than $0^m.4$ and should have a declination in the range $90 - \phi \pm 30$.

12 Supporting tools

12.1 Daemon shutdown

This supporting program is a daemon that runs on the machine `druid` at startup and performs one single task: turn off the machine by the command received via a TCP/IP connection. To prevent a situation where one user (the program `ameba`, for example) may terminate the operation of the machine regardless of other users, a daemon `shutdown` handles the locking requests. Once user no longer needs for the machine, he should withdraw his code from the lock. The program requires no initialization and parking and handles three commands:

- **run shutdown** — start turning off the machine. Performed only in the absence of locks.
- **set lock=.....** — setting the lock on. Response to the command shows the current number of the installed locks. Argument is a character string of any length: a key of this lock.
- **set unlock=.....** — unlock. Response to the command shows the current number of remaining locks. To successfully execute the command argument must coincide with the key lock.
- **get lock** — query the number of installed locks.

To ensure that the program `shutdown` could switch off the machine, while maintaining local security, a special setting for utility `sudo` is used allowing the user `mass` to run the system utility `/sbin/shutdown`.

12.2 Time synchronization

On all computers time synchronization is made by the conventional method using constantly running `ntpd` daemon of `ntp` protocol. It uses the following nodes of the `ntp` service:

- `ru.pool.ntp.org`
- `europe.pool.ntp.org`
- `pool.ntp.org`

For real delays (about 700 ms) in the satellite communication channel the time on the ASM machines is known usually with an accuracy better than $20 \div 30$ ms.

[6] See `http://curl/mass/download/doc/new_mass_cat.pdf`

12.3 Web–cameras

Two survey web-cameras (`dome` and `yard`) attached to the machine `omicron` are served by the `camsource 0.7.0` (`http://camsource.sourceforge.net`), which is a daemon for capture video images and a simple http-server. Although this program is not supported for several years and contains some mistakes, it works fine as a daemon minimally loading the machine. The usb-controller bandwidth on the machine `omicron` is not enough to fully work with two cameras, so the driver for Philips Camera `pwc` runs at maximum compression.

CGI-script `eremaea`, running on the machine `eagle`, each minute requests images from both cameras via http-protocol and stores them in an appropriate archive. Snapshots is stored for 5 days. On the expiry date the image files are removed by the daemon. Access to images is possible at `http://eagle/eremaea`.

12.4 Data archiving

For various reasons, the standard tool `logrotate` is not suitable for automatic backup of the output files of the ASM software components. This problem is solved with a script `rotate.sh` posted on each machine of the monitor and the CGI-script `pimenta` operating at the backup machine `eagle`. Once a day daemon `cron` launches the script `rotate.sh` which finds among the output files those that did not vary since more than a day, compresses them with help of `bz2` utility and sends to the server `eagle` using the utility `cUrl` by PUT-method of http-protocol, where `pimenta` puts these files in the directories.

12.5 Website http://eagle.sai.msu.ru

For public access to information about current conditions on the ASM, the dedicated website was developed. The program to generate pages for the site is written in C++ (CVS package www_eagle) and works via FastCGI — client-server protocol for interaction between a web server and some application. For the corresponding settings of the http-server Apache, the package has needed configuration files. Data needed to display information are collected by a special daemon `sensord`, which uses the protocol SNMP (Simple Network Management Protocol) for this. The following network nodes provide these data:

- two access points (data of input/output traffic)
- omicron (requested for the `monitor` information)
- eagle (data from the sensor of clear sky that are requested from the program `everest`)

Converting data in SNMP is performed by special binary dynamic modules for `net-snmpd`: `rainAgent` and `meteoAgent`. Widely known package RRD (Round Roubin Databases) [7] is used for graphics. Daemon `sensord` adds information to the RRD database.

Fig. 6 shows an example of the page ‘‘`Weather`’’ at the ASM web-site. In between of 21 and 24 hours of Wednesday the lower curve on the chart illustrates the temperatures described in the text of a fogging of the humidity sensor, resulting in unrealistically low values of RH and Dew point.

[7] `http://oss.oetiker.ch/rrdtool/`

13 Appendix A

Table 1: Background information on software components

Component	Machine	Local IP	Port	CVS module
`ameba`	`eagle`	192.168.10.8	—	ameba
`dome`	`omicron`	192.168.10.5	16302	dome
`monitor`	`omicron`	192.168.10.5	16300	monitor
`tlsp`	`druid`	192.168.10.4	16400	rcx400_scope
`dimm`	`druid`	192.168.10.4	16200	dimm_tool
`mass`	`druid`	192.168.10.4	16201	turbina_core_m
`shutdown`	`druid`	192.168.10.4	16100	shutdown_daemon

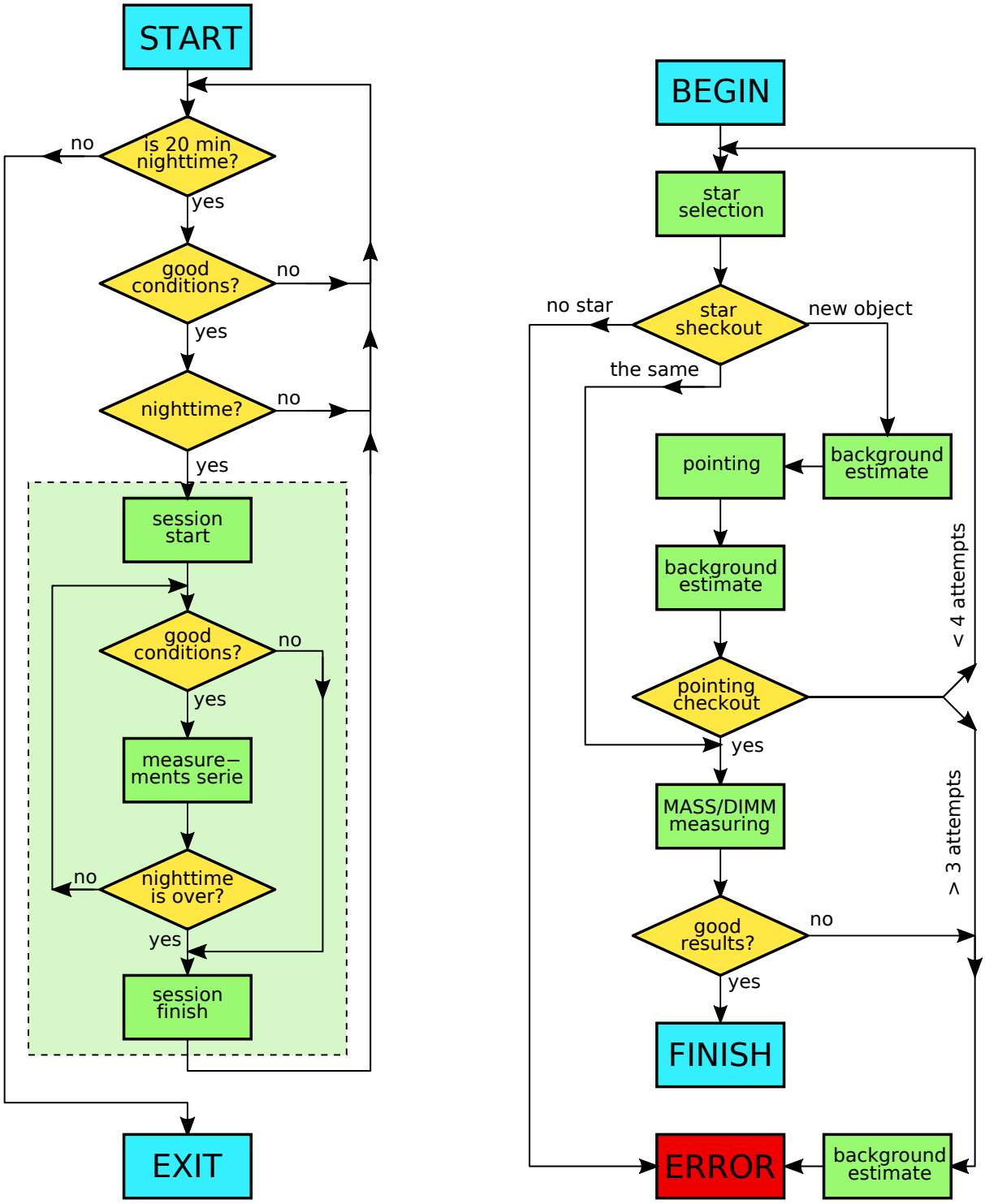

Figure 5: Diagram of operation of `ameba` program

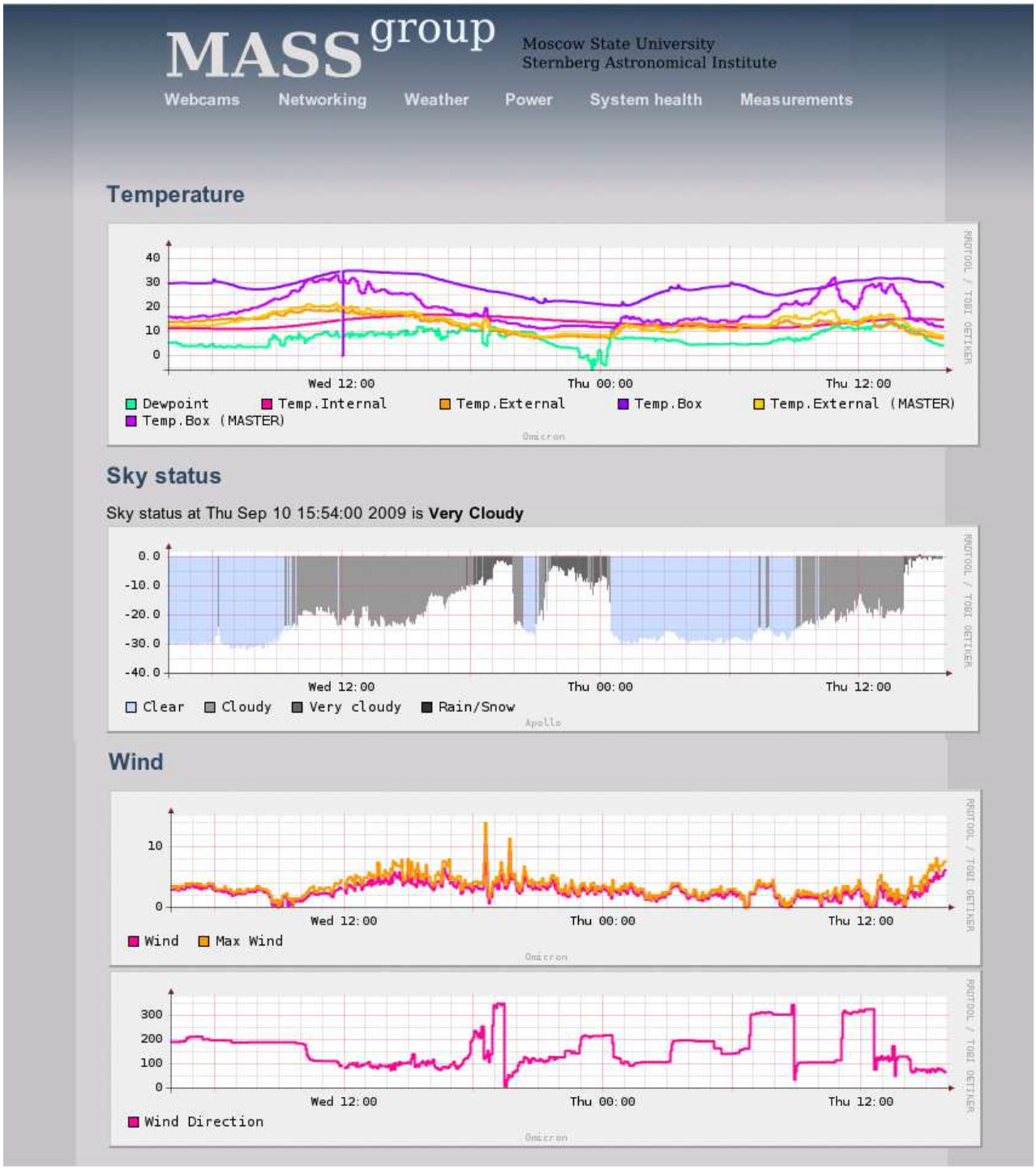

Figure 6: Appearance of the page ‘‘`Weather`’’ of the web-site `http://eagle.sai.msu.ru`. From top to bottom: 6 temperature dependencies measured at different locations, the behavior of the temperature of the sky: gray – cloudy, blue – clear; wind speed and direction.