Комплекс управляющих программ автоматического астроклиматического монитора

В. Корнилов, О. Возякова, М. Корнилов, Б. Сафонов, Н. Шатский

14 октября 2009 г.

Введение

Автоматический астроклиматический монитор (АСМ[1]) ГАИШ установлен на вершине горы Шатджатмаз в 40 м к юго-западу от отметки центра башни 2.5 м телескопа согласно проекту Кавказской горной обсерватории ГАИШ МГУ. Монитор был установлен и запущен в 2007 г с целью сбора важнейших астроклиматических характеристик: статистики качества изображения, статистики высотного распределения оптической турбулентности, количества ясного ночного времени, данных о прозрачности атмосферы, и самых необходимых метеохарактеристик — скорости и направления приземного ветра, суточного и сезонного хода температуры окружающего воздуха и влажности.

Полученная информация будет использована для полного астроклиматического описания места установки 2.5 м телескопа, оценки перспективности реализации системы адаптивной оптики, стратегического планирования наблюдательных задач с целью всемерного повышения эффективности работы телескопа. В дальнейшем текущие данные, получаемые с помощью АСМ, будут использоваться для оперативной оптимизации расписания наблюдений на телескопе.

1 Информационная структура АСМ

Автоматический астроклиматический монитор включает в свой состав следующие вычислительные и сетевые средства:

1. Машина `eagle` — полное имя `eagle.sai.msu.ru`, локальный IP 192.168.10.8. Установлена в помещении узла связи ГАС ГАО. Эта машина используется как входной шлюз астроклиматического монитора при внешнем доступе. На машине работает http-сервер для отображения текущего состояния АСМ. На ней же организовано архивное хранение данных измерений, поступающих с других машин системы, и изображений с обзорных WEB-камер.

2. Точки доступа 192.168.10.2 и 192.168.10.3 обеспечивают сетевое соединение с машинами, установленными на вышке монитора. Первая точка расположена в здании оптической лаборатории ГАС ГАО, вторая — на вышке монитора примерно в 800 м от первой на вершине горы.

3. Машина `omicron` (локальный IP 192.168.10.5) установлена под куполом монитора в металлическом шкафу. Ее назначение — постоянный сбор метеоданных, контроль и управление основным питанием АСМ, управление куполом, получение изображений с внешней (yard) и внутренней (dome) обзорных камер (SPC900NC Philips), подключенных к ней через USB интерфейс. К последовательному порту через специальный конвертер COM/RS485 подключена линия RS485 для обмена данными и командами с контроллером питания, купола и метеоконтроллером. Машина постоянно включена, однако в период с 11:57:00 по 12:03:00 местного времени она автоматически перезагружается и перезапускает все необходимые программные компоненты.

[1] Аббревиатура АСМ используется по аналогии с названием комплекса ASM (Astronomical Site Monitor) Европейской южной обсерватории

4. Машина `druid` (IP 192.168.10.4) установлена там же, где и `omicron`, включается в начале сессии измерений, останавливается и выключается при завершении сессии измерений. К этой машине подсоединены телескоп Meade RCX400, прибор MASS/DIMM и камера искателя/гида. Ее назначение — управление телескопом RCX400, управление и сбор данных с каналов DIMM и MASS прибора MASS/DIMM. Соответственно, обмен с камерой искателя/гида (SPC900NC Philips) осуществляется через USB порт, камера DIMM-канала (Prosilica EC650) подключена через IEEE-1394 интерфейс, канал MASS подсоединен через конвертер LPT/RS485 к параллельному порту, и телескоп RCX400 и его контроллер питания — к последовательному порту.

Структура АСМ схематически представлена на Рис. 1, где слева изображена аппаратура, установленная в помещениях ГАС ГАО, а справа — аппаратура, установленная на вышке АСМ и в рядом расположенном вагончике (контроллер метеослужбы и его датчики на мачте от ветрогенератора).

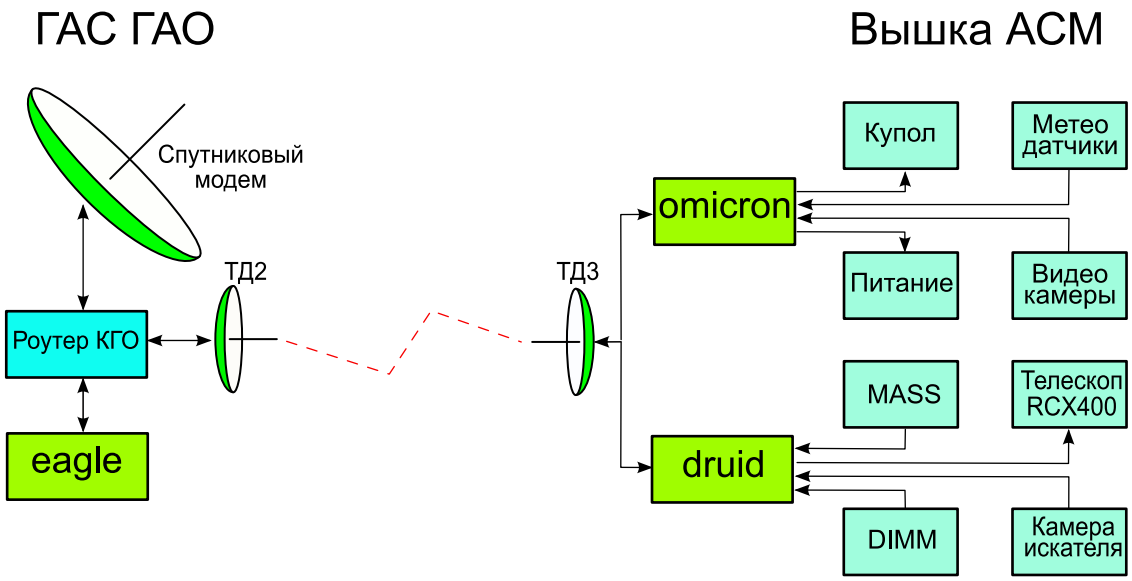

Рис. 1: Аппаратная схема автоматического астроклиматического монитора ГАИШ

2 Состав программного обеспечения АСМ

Работа астроклиматического монитора в настоящее время обеспечивается следующими программными компонентами:

- `tlsp` — программа управления телескопом RCX400 и камерой искателя;
- `dimm` — программа управления и сбора данных DIMM-канала прибора;
- `mass` — программа управления и сбора данных MASS-канала прибора;
- `shutdown` — демон удаленного выключения машины `druid`;
- `dome` — программа управления куполом и питанием измерительной аппаратуры;
- `monitor` — программа сбора метеоинформации и информации о системе энергоснабжения;

– `ameba` — программа управления процессом измерений.

Данные о состоянии неба запрашиваются у программы `everest`, работающей на сервере системы МАСТЕР `apollo.sai.msu.ru`, порт 15012.

Отображение текущего состояния АСМ и другой информации через WEB интерфейс осуществляется компонентами, работающими на машинах `eagle` и `omicron`:

1. `meteoAgent` — трансляция данных программы `monitor`,
2. `rainAgent` — трансляция данных программы `everest`.

3 Принципы построения программного обеспечения

Полученный в 2002 – 2006 гг опыт разработки и эксплуатации программного обеспечения прибора MASS/DIMM привел к изменениям первоначальных подходов и требований к программному обеспечению, ориентированному на автоматическую работу АСМ и других подобных автоматических комплексов. Подробное обсуждение приведено в документе "Новое программное обеспечение прибора MASS/DIMM. Управляющее ядро Turbina-core" [2], где также выработаны основные требования к ПО и его элементам. В целом, в процессе проектирования и программирования мы придерживались этих принципов, хотя некоторые частные представления значительно эволюционировали.

Итак, при разработке основного программного обеспечения мы старались, где это возможно, придерживаться следующих подходов:

1. Программное обеспечение состоит из нескольких отдельных программных компонентов, которые запускаются и работают независимо друг от друга и при необходимости взаимодействуют друг с другом. Разделение всего программного обеспечения на отдельные компоненты позволяет 1) существенно повысить гибкость всей системы, переложив часть функций на операционную систему, 2) упростить процесс разработки и отладки программного обеспечения, 3) обеспечить устойчивость всей системы в целом, 4) упростить логику работы отдельного программного компонента, 5) органично обеспечить параллельное выполнение различных функций.
2. Каждый программный компонент обеспечивает выполнение одной логической задачи или функции АСМ. Это подразумевает разделение всей работы на отдельные, достаточно изолированные и самодостаточные части. При этом приоритет отдается именно задаче, а не обслуживанию того или иного устройства. Например, программа `tlsp` отвечает за наведение телескопа на нужный объект, его центрирование и гидирование на протяжении времени измерения. Для этого программа взаимодействует с телескопом, контроллером питания телескопа и камерой искателя. С другой стороны, контроллер купола используется двумя программами — `dome` и `monitor`.
3. Фоновый режим работы программы является основным, поэтому пользовательский интерфейс отсутствует. Однако, по умолчанию компоненты стартуют в приоритетном режиме (foreground). Перевод в фоновый режим при необходимости осуществляется самой программой, а не запускающей оболочкой. Такой подход был выбран, чтобы

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

на фазе старта программы иметь расширенную информацию о неуспешном запуске, а затем полностью освободить стандартные каналы ввода-вывода. Таким образом, при штатной работе все программные компоненты функционируют как “демоны” (daemon).

4. Сообщения о ходе работы и ошибках в обязательном порядке записываются в файл протокола работы программы. Некоторые программные компоненты не “производят” выходных данных (например, `ameba`), в таких случаях протокольный файл должен содержать всю информацию о работе программы. Имена протокольных файлов формируются однотипным образом `YYMMDD-program.log`, где `YYMMDD` — дата старта программы (календарная дата вечера ночи наблюдений, смена даты в местный полдень), `program` — имя программы[3].
5. Управление работой программы осуществляется 1) при помощи ключей в командной строке при запуске, 2) при помощи данных конфигурационного файла и 3) командами, поступающими через сетевые соединения. Ключи имеют одинаковый смысл для всех программ, поскольку обрабатываются общим модулем и относятся к контексту работы программы. Конфигурационные файлы также имеют одинаковую структуру, но содержат параметры настройки функциональной части работы программы и внешних устройств. Команды, поступающие через сетевые соединения, инициируют конкретное единичное действие программы. Настроечные параметры, как правило, этими командами не изменяются.
6. Формат команд для оперативного управления описан в “SUPERVISOR program User Guide SV version 0.24”[4] и применялся ранее в программном обеспечении MASS и других проектов. Использование текстовой команды регламентированного формата позволяет эффективно осуществлять ручное управление, что существенно на этапе отладки системы в целом.
7. Синхронизация работы осуществляется на командном уровне. Роль **супервизора**, реализующего общий алгоритм функционирования АСМ, выполняет программа `ameba`. Остальные программные компоненты функционируют как серверы. Имеется только одно исключение: программа `tlsp` использует для автоматического гидирования информацию канала DIMM — величину смещения звезды относительно центра рабочей апертуры. Для этого она самостоятельно как клиент посылает запросы на выполнение соответствующего измерения программе `dimm`. Поскольку такое измерение требует 2 с, то синхронизация осуществляется по принципу: удовлетворяется пришедший раньше запрос. Пришедшая позже команда (получившая в ответ `BUSY`) при необходимости повторяется через небольшой промежуток времени.
8. Логическая структура программных компонентов строится на общей основе. Одинаковые функции внутри программ обеспечиваются общими модулями. Прежде всего это: поддержка конфигурирования, обработка ошибок, сетевой обмен, разбор внешних команд, обмен по линии RS485, время и астрономические координаты. Структура каждой программы содержит статическую часть и динамически создаваемую в активном (инициализированном) состоянии часть — так называемое устройство (Device). Статическая часть поддеживает общие функции, создание устройства при инициализации и уничтожение при парковке.

[3]Протокольный файл программы `monitor` содержит обозначение `sens`

[4]см. http://dragon.sai.msu.ru/mass/download/doc/sv_ug.pdf

9. Программные компоненты в основном реализованы на языке C++ с активным использованием стандартной библиотеки шаблонов. По возможности поддерживался общий стиль программирования. Исходные коды в обязательном порядке хранятся и развиваются в системе CVS. Все программное обеспечение предназначено для работы под операционной системой GNU/Linux. Предпочтительный дистрибутив — OpenSuse 9.2, 10.2, 11.2.

Перечисленные подходы к разработке и поддержке программного обеспечения ACM соответствуют современным идеям шаблонного проектирования и позволили разработать и отладить весь набор компонентов с минимальной затратой ресурсов.

4 Размещение, запуск и останов программных компонентов

Каждый программный компонент `program` размещен в соответствующем каталоге `/opt/program/`. Структура таких каталогов унифицирована. Имееются следующие поддиректории:

- `etc/` для конфигурационных файлов,
- `data/` для всех входных и выходных файлов,
- `data/log/` для файла протокола работы,
- `data/out/` для файла основных выходных данных.

Исполняемый файл находится в корневом каталоге `/opt/program/`. Могут быть также дополнительные поддиректории, например, `/opt/dimm/data/images/` предназначена для изображений, получаемых по запросу или в качестве протокольных кадров с камеры DIMM-канала.

Владельцем файлов и каталогов является системный пользователь `mass` группы `mass`, обладающий необходимыми привилегиями. В случае отладочной или экспериментальной работы программы могут быть запущены любым пользователем от лица `mass`, для чего атрибуты исполняемого файла содержат установленный s-бит.

В штатном режиме программные компоненты запускаются системной программой `cron` либо на старте компьютера (`tlsp`, `dimm` и `mass` на машине `druid`, `monitor` на машине `omicron`), либо в назначенное время (`dome` на машине `omicron`, `ameba` на машине `eagle`)

В зависимости от ситуации, запуск программного компонента может осуществляться с различными опциями (ключами) в командной строке. Приведенные ниже ключи являются общими для всех программных компонентов:

`-a` — инициализация программы и оборудования на старте;

`-b` — переход в фоновый режим после старта;

`-c name` — использовать конфигурационный файл `name` вместо файла по умолчанию;

`-d` — запуск программы в отладочном режиме;

`-p N` — использовать номер порта `N` вместо номера по умолчанию;

`-i host` — использовать `host` как имя машины вместо `0.0.0.0` по умолчанию.

В штатном режиме останов программ осуществляется командами `quit`, подаваемыми программой `ameba` каждому компоненту. Для того, чтобы минимизировать риск повреждения аппаратуры при отказах сетевых соединений, на каждой машине предусмотрен запуск

сторожевых скриптов, заведомо после завершения наблюдений, но до восхода Солнца. Их главное назначение — выключить питание измерительной аппаратуры (в том числе высокое напряжение в MASS-канале) и закрыть купол.

Ручное управление всеми программными компонентами осуществляется с использованием сетевых системных программ, таких как `telnet` или `netcat`. При использовании `telnet` достаточно соединиться с нужным портом на той машине, где запущен компонент, и подавать текстовые команды в соответствии с используемым протоколом. Например, для соединения с программой `tlsp` достаточно с консоли любой машины, находящейся в локальной сети АСМ, выполнить команду `telnet druid 16400`, где 16400 — номер порта по умолчанию. Если программа `tlsp` запущена, соединение будет установлено, и любая синтаксически правильная команда будет принята к выполнению. Утилиту `netcat` удобно использовать в скриптах.

5 Общие программные модули и команды

Как уже было упомянуто, общие логические функции программ обслуживаются общими программными модулями. По разным причинам эти модули используются в виде исходных файлов, размещенных в каталогах `infras` и `server`, а не в виде объектной библиотеки. Модули обеспечивают 5 основных функции:

- конфигурирование программного компонента;
- обмен по протоколу TCP/IP с другими программами или пользователем;
- синтаксический разбор внешних команд и обеспечение их выполнения;
- обмен с оборудованием по интерфейсу RS485;
- обработка ошибок и запись протокола работы.

Структура конфигурационных файлов сохранена такой же, как и у программы `Turbina`: это набор отдельных озаглавленных секций, выделенных ключевыми словами `Section` и `EndSection`, и содержащих некоторый логически объединенный набор подсекций `SubSection`. Каждая подсекция содержит конкретные конфигурируемые параметры, относящиеся к отдельному объекту программы. Например, в конфигурационном файле программы `dimm` секция `General` содержит две подсекции: `Site` и `DIMM`. Первая из них содержит параметры, определяющие место работы прибора:

```
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
```

Строка параметра состоит из его имени и значения. После чтения конфигурационного файла значение параметра хранится в программе как элемент трехмерного ассоциативного массива. Не все параметры, содержащиеся в конфигурационном файле, необходимы самой программе, эти параметры просто переписываются в выходной файл и затем используются

обрабатывающим ПО. Такой подход является отражением принципа, что выходные данные обязаны содержать всю необходимую для обработки и последующей интерпретации информацию.

Конфигурационный файл в какой-то мере повторяет логическую структуру программы, поскольку параметры естественным образом объединяются в секции и подсекции, соответствующие программным объектам. Расположение некоторых параметров достаточно условно, например, приведенный выше `DeviceNumber` может находиться и в другом месте файла, поскольку не используется ни самой программой, ни обрабатывающим ПО.

Обмен командами и данными через сетевые соединения в виде текстовых сообщений обеспечивается модулем `server`. Признаком окончания текстовой команды является терминальный символ `<CR>`. Функционирование модуля слабо связано с принятым командным протоколом и ориентировано на правильное обслуживание присоединений и отсоединений сетевых клиентов и обработка ошибочных ситуаций. Кроме серверной части, в модуле содержится также класс, на котором основана клиентская часть. Напомним, что обмен происходит на строго индивидуальной основе — ответ на команду или запрос будет направлен только запросившему действие или информацию клиенту.

Модуль синтаксического разбора команд `comms` обеспечивает поддержку командного протокола. Через соответствующие интерфейсные прослойки он производит инициализацию (команда `init`), парковку программы (команда `park`), завершение программы (команда `quit`), извлечение актуальных данных (команда `get`), модификацию программных параметров или процессов (команда `set`) и, собственно, выполнение основных функций программы по команде `run`. Одна из основных функций модуля — синхронизация выполнения этих команд.

Немодифицирующая состояние программного компонента команда `get` выполняется всегда. Другие команды будут выполнены, если программа инициализирована. Отличие обработки команды `set` от команды `run` заключается в том, что первая выполняется на временах заведомо меньших характерного тайм-аута сетевого обмена, а команда `run` требует большего времени. Поэтому она (также как и команды `init` и `park`) запускается в специальном программном потоке.

Инициализация программы состоит из следующих этапов: чтение конфигурационного файла, создание программных объектов, открытие файлов выходных данных, соединение с аппаратурой и ее инициализация. После успешной инициализации программа переходит в состояние `READY`.

Парковка производится в обратном порядке: отключение аппаратуры, закрытие обмена и файлов выходных данных. После парковки программа переходит в состояние `PARKED`. Также парковка производится при выполнении команды `quit`.

Ни одна команда типа `set` или `run` не будет выполнена, если программный компонент (и соответствующее оборудование) находится в запаркованном состоянии. В этом состоянии возможно только получить информацию об общем состоянии программы, для чего модуль разбора команд самостоятельно обеспечивает 3 команды типа `get`:

`get status` — возвращается состояние компонента: `OK STATUS=READY`, `OK STATUS=BUSY` или `OK STATUS=PARKED`.

`get ident` — возвращается название программы и ее текущая версия в виде, например, `IDENT= ''Turbina-core(D) Vers. 2.26''`.

`get error` — возвращается описание последней случившейся ошибки в формате выдачи в файл протокола.

В дальнейшем при описании команд конкретных программ эти команды, также как и комады `init`, `park`, `quit`, будут опускаться.

Взаимодействие программ с разработанной нами аппаратурой осуществляется при помощи пакетного обмена по физическому интерфейсу RS485. Для поддержки этого взаимодействия, наряду с разработанным нами соответствующим системным драйвером, используется набор программных модулей. Интерфейсные модули обеспечивают все необходимые функции обмена, включая обработку ошибок и эмуляцию обмена для отладочных целей. Также они обеспечивают синхронизацию (взаимоблокировку) обращений из разных программных потоков.

Принятый способ обработки ошибочных ситуаций, возможных при работе программных компонентов, основан на механизме генерации исключений в месте обнаружения ошибки и трансляции его в верхние программные слои, где ошибки обрабатываются или просто фиксируются в файле протокола. Кроме сообщений об ошибках, в файл протокола направляются сообщения об основных событиях в работе программы. Расширенная выдача в файл протокола включается при запуске программы с опцией `-d` в командной строке. Как правило, в программах, взаимодействующих с оборудованием по RS485 интерфейсу, протоколируется весь обмен управляющими командами. В `log`-файле программы `ameba` фиксируются все посланные сетевые команды и полученные ответы.

Структура типичного сообщения в файле протокола иллюстрируется следующими строками из протокола программы `dimm`

```
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
```

После момента времени (UT) в скобках приводится код ошибки, предназначенный для автоматической обработки ошибок клиентом и для фильтрации протокольных сообщений (поиск одинаковых или близких ошибок). Код 000 обозначает информационное сообщение, а не ошибку. Текстовое описание ошибки состоит из двух частей, заполнение которых возможно на разных логических уровнях программы для более простой локализации источника ошибки.

6 Программа mass

Программа выполняет сбор и первичную обработку данных, поступающих с MASS-канала прибора MASS/DIMM, и управляет этим каналом. Принципы измерений аналогичны использованным ранее в программе Турбина[5]. Сбор и обработка осуществляется на следующих временны̀х интервалах:

– `exposition` – время элементарного измерения светового потока (микроэкспозиция), задается в миллисекундах

[5]http://dragon.sai.msu.ru/mass/download/doc/main_document.pdf

– `basetime` – интервал (в секундах), на котором вычисляются статистические моменты относительных флуктуаций потоков
– `accumtime` – длительность измерительной моды (в секундах)

Все выдаваемые программой значения потоков, приводятся к длительности микроэкспозиции.

В программе реализованы следующие моды, то есть специализированные функции измерения:

– нормальная мода (Normal mode) – измерение мерцаний звезд. Время выполнения определяется параметром `Normal mode/Accumtime`
– измерение фона (Background measurement) – измерение фона неба в течение `Background measurement/Accumtime`.
– оценка потока (Flux estimation) – вспомогательное измерение потока в течение `Tests/FluxEstimatior` обычно используется при наведении на объект.
– тест детекторов (Detector test – измерение контрольного источника света.
– статистический тест (Statistic test) – измерение искусственых мерцаний.
– тест обмена (Exchange test) – проверка наличия сбоев передачи данных с прибора в компьютер.

Последовательность мод может быть задана в виде сценария — комбинации кодов мод и арифметических операций. Пример: F+20*(N+B), где F – оценка потока, N – нормальная мода, B – измерение фона. В этом примере сначала выполняется оценка потока, затем двадцать раз повторяется нормальная мода и измерение фона.

Список поддерживаемых команд:

– **run** или **run normal** – запуск нормальной моды
– **run background** – измерение фона
– **run flux** – оценка потока
– **run dtest** – проверка приемника
– **run stest** – статистический тест
– **run etest** – тест обмена
– **run normal background dtest** – последовательный запуск нескольких мод
– **run scenario=2*N+F** – запуск сценария 2*N+F
– **run scenario** – запуск предварительно установленного сценария
– **get flux** – запрос потока, возвращается поток в четырех апертурах `FluxA=... FluxB=... FluxC=... FluxD=...`, усредненный по последней завершившейся моде
– **get data** – запрос результатов, возвращается строка результатов последней завершившейся моды (в формате выходного файла)
– **get temperature** – запрос температуры внутри прибора
– **get hv** – запрос состояния высокого напряжения. Возвращается `ON` или `OFF`.
– **get illum** – запрос состояния подсветки диафрагмы подсмотра. Ответ `ON` или `OFF`.
– **get light** – запрос состояния контрольного источника света. Ответ `ON` или `OFF`.

– **get mirror** – запрос положения зеркала подсмотра. Ответ `ON` — зеркало перебрасывает свет в окуляр подсмотра, `OFF` — зеркало направляет свет на детекторы.
– **set scenario=2*N+F** – установка сценария 2*N+F
– **set object="..."** – запись в выходной файл информации об объекте измерений, например: `20 set object=''7949 Eps_Cyg 20 46 13 +33 58 13 2.46 1.03 K03''`
– **set illum=on/off** – включение/выключение подсветки диафрагмы
– **set hv=on/off** – включение/выключение высокого напряжения ФЭУ
– **set cnt=on/off** – включение/выключение записи отсчетов единичных экспозиций (`exposition`) в *.cnt-файл
– **stop** – остановка сценария после завершения текущей моды
– **stop now** – прерывание измерения после завершения текущего `basetime`

Результаты `basetime`-измерений выдаются в выходной файл `yymmdd-mass.stm` в строках с префиксом "m". Строки результатов, усредненных по `accumtime`, имеют префикс "A". Формат строк одинаков и содержит следующие поля:

1. Префикс
2. Дата окончания измерения в формате YYYY-MM-DD
3. Момент окончания измерения, приводится UT в виде hh:mm:ss
4. Средний поток в канале A
5. Средний поток в канале B
6. Средний поток в канале C
7. Средний поток в канале D
8. Дисперсия относительных флуктуаций потока в A
9. Дисперсия относительных флуктуаций потока в B
10. Дисперсия относительных флуктуаций потока в C
11. Дисперсия относительных флуктуаций потока в D
12. Дисперсия разности относительных флуктуаций в каналах A и B
13. Дисперсия разности относительных флуктуаций в каналах A и C
14. Дисперсия разности относительных флуктуаций в каналах A и D
15. Дисперсия разности относительных флуктуаций в каналах B и C
16. Дисперсия разности относительных флуктуаций в каналах B и D
17. Дисперсия разности относительных флуктуаций в каналах C и D
18. Ковариация с лагом 1 относительных флуктуаций потока в A
19. Ковариация с лагом 1 относительных флуктуаций потока в B
20. Ковариация с лагом 1 относительных флуктуаций потока в C
21. Ковариация с лагом 1 относительных флуктуаций потока в D
22. Ковариация с лагом 1 разности относительных флуктуаций в каналах A и B
23. Ковариация с лагом 1 разности относительных флуктуаций в каналах A и C
24. Ковариация с лагом 1 разности относительных флуктуаций в каналах A и D
25. Ковариация с лагом 1 разности относительных флуктуаций в каналах B и C

26. Ковариация с лагом 1 разности относительных флуктуаций в каналах B и D
27. Ковариация с лагом 1 разности относительных флуктуаций в каналах C и D
28. Ковариация с лагом 2 относительных флуктуаций потока в A
29. Ковариация с лагом 2 относительных флуктуаций потока в B
30. Ковариация с лагом 2 относительных флуктуаций потока в C
31. Ковариация с лагом 2 относительных флуктуаций потока в D

Пример `m`-строки

```
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
```

7 Программа dimm

Режимы работы, форматы выходных файлов и команды программы `dimm` подробно описаны в специальном документе "Turbina-core(D): Dimm User Guide"[6].

Так же, как и в программе `mass`, используются следующие временны̀е интервалы:

- `exposition` – экспозиция кадра ПЗС-камеры (в миллисекундах).
- `basetime` – интервал (в секундах), на котором вычисляются параметры дрожаний и характеристики изображений звезд
- `accumtime` – длительность измерительной моды (в секундах).

Ниже приводится список поддерживаемых команд с кратким пояснением смысла выполнемого действия.

- **run** или **run normal** – запуск основной моды измерения дифференциальных дрожаний в рабочем окне
- **run center** – запуск измерения характеристик изображений в окне, охватывающем всю диафрагму поля
- **run test** – запуск измерения освещенной апертуры для определения параметров прибора
- **run scenario=2*N+C** – запуск сценария измерений 2*N+C
- **run raw** – запуск основной моды с выводом результатов отдельных измерений
- **run estimation** – запуск основной моды измерения продолжительностью в один `basetime`
- **run pictures** – запуск записи в файл последовательности изображений рабочего окна
- **get offset** – возвращает текущее смещение общего центра тяжести изображений относительно центра апертуры поля (в пикселах).
- **get separation** – возвращает разделение изображений по x и y (в пикселах).
- **get flux** – возвращается полная и максимальная интенсивность левого и правого изображения (в отсчетах за экспозицию)

[6] `http://dragon/mass/download/doc/dimm_soft_description.pdf`

– **get data** – запрос результатов, возвращается строка результатов последней завершившейся основной моды (в формате выходного файла)
– **get mode** – запрос названия текущей или последней выполненной моды
– **set scenario=2*N+C** – установка сценария 2*N+C
– **set object=”...”** – запись в выходной файл информации об объекте измерений, например: `20 set object=‘‘6396 Zet_Dra 17 08 47 +65 42 53 3.17 -0.12 B55’’`
– **stop** – остановка сценария после завершения текущей моды
– **stop now** – прерывание измерения после завершения текущего `basetime`

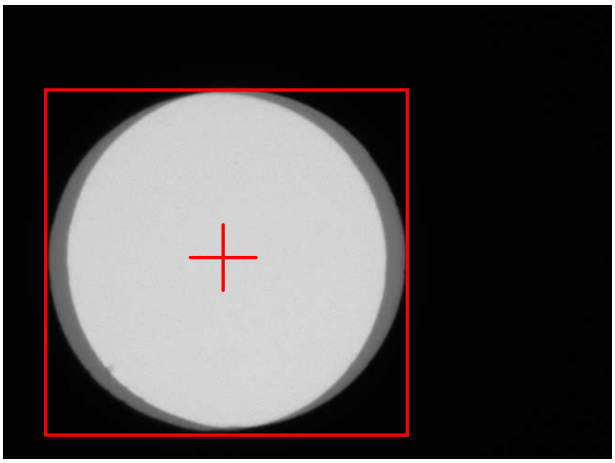

Рис. 2: Вид апертуры поля на кадре, полученном по команде **run test**

Результаты `basetime`-измерений выдаются в выходной файл `yymmdd-dimm.stm` в строках с префиксом "d". Строки результатов, усредненных по `accumtime`, имеют префикс "D". Формат строк одинаков и содержит следующие поля:

1. Префикс.
2. Дата в формате YYYY-MM-DD.
3. Момент окончания измерения, приводится UT в виде hh:mm:ss.
4. Число полученных и обработанных кадров рабочего окна.
5. Полный поток в левом изображении, отсчеты.
6. Полный поток в правом изображении, отсчеты.
7. Среднеквадратичное нормированное значение флуктуаций потока в левом изображении.
8. Среднеквадратичное нормированное значение флуктуаций потока в правом изображении.
9. Максимальный сигнал в левом изображении, отсчеты.
10. Максимальный сигнал в правом изображении, отсчеты.
11. Среднее разделение между изображениями по x координате, пикселы.
12. Среднее разделение между изображениями по y координате, пикселы.

13. Среднеквадратичное значение флуктуаций разделения по x, пикселы.
14. Среднеквадратичное значение флуктуаций разделения по y, пикселы.
15. Ковариация флуктуаций разделения по x с лагом 1, пиксел^2.
16. Ковариация флуктуаций разделения по y с лагом 1, пиксел^2.
17. Оценка среднеквадратичного значения шума по x, пикселы.
18. Оценка среднеквадратичного значения шума по y, пикселы.
19. Среднее положение общего центра тяжести изображений по x, пикселы.
20. Среднее положение общего центра тяжести изображений по y, пикселы.
21. Среднеквадратичное значение среднего положения по x-кооридинате, пикселы.
22. Среднеквадратичное значение среднего положения по y-кооридинате, пикселы.
23. Среднее значение FWHM левого изображения, пикселы.
24. Средняя эллиптичность левого изображения.
25. Среднее значение FWHM правого изображения, пикселы.
26. Средняя эллиптичность правого изображения.
27. Среднее значение фона, отсчеты.
28. Среднеквадратичное значение флуктуаций фона, отсчеты.

Пример `d`-строки:

```
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
```

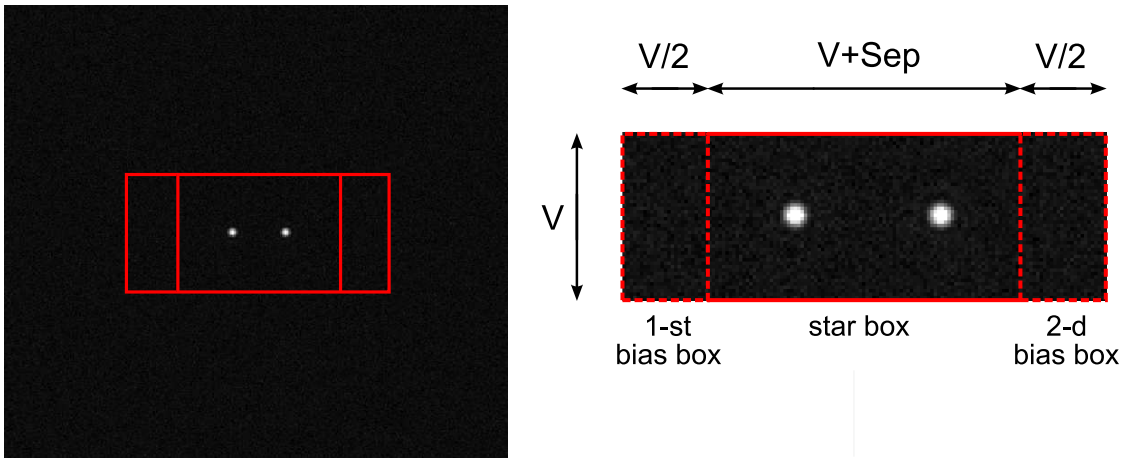

Рис. 3: Слева: изображения звезды на кадре, полученном по команде **run center**. Справа: структура рабочего окна, используемого в основой моде измерений

8 Программа tlsp

Программа `tlsp` предназначена для управления 30-см телескопом Meade RCX400, подсоединенным к машине `druid` через последовательный порт (устройство `/dev/ttyS0`). Помимо этого, программа обеспечивает поиск, центрирование и гидирование программных звезд по изображениям, получаемых с камеры искателя. При гидировании по каналу DIMM, программа работает как TCP/IP клиент, периодически запрашивая смещение изображений звезды относительно оптической оси прибора. Такое отступление от основного принципа — программные компоненты взаимодействуют через единый центр, выполняющий роль супервизора (программа `ameba`) — допущено для инкапсуляции функции гидирования внутри одной программы.

Список команд программы `tlsp`:

- **run ra[**=...**]** или **run dec[**=...**]** или **run ra[**=...**] dec[**=...**]** – запуск наведения телескопа в экваториальной системе координат. Если аргументы в команде отсутствуют, используются значения, установленные ранее командой **set**. Аргументы подаются в строковом виде в обычном формате (см. пример ниже). Наведение включает запуск часового ведения, поиск ярчайшей звезды в окрестности точки наведения (приблизительно $3^\circ \times 2^\circ$) и грубое центрирование.
- **run az[**=...**]** или **run alt[**=...**]** или **run az**=... **alt**=... – наведение телескопа в астрономической горизонтальной системе координат (см. пример ниже). В отличие от наведения в экваториальной системе координат, поиск звезды не производится, а часовое ведение выключается.
- **run point** – запуск наведения телескопа в экваториальной системе координат в предустановленную командой `set` точку.
- **run center** – запуск тщательного центрирования звезды в поле искателя.
- **run search** – запуск поиска звезды в окрестности 3×3 поля зрения искателя (приблизительно $3^\circ \times 2^\circ$) и последующее грубое центрирование.
- **run dx**=... **dy**=... – смещение телескопа в горизонтальной системе координат в секундах дуги.
- **run focus**=**shift** – фокусировка телескопа, где аргумент `shift` представляет собой относительное смещение от текущей позиции в некоторых условных единицах.
- **stop** – остановка наведения или любой команды **run**.
- **set guide**=**on** или **set guide**=**dimm** – включение гидирование телескопа по данным камеры искателя или DIMM-канала
- **set guide**=**off** – выключение гидирования
- **set sync**=**on** – приведение системы координат телескопа в соответствие с координатами последней цели. Эта команда полезна для уточнения системы координат в случае, если центрирование звезды потребовало значительного смещения телескопа
- **get ra dec** – запрос текущего положения телескопа в экваториальной системе координат.
- **get az alt** – запрос текущего положения телескопа в горизонтальной системе координат.
- **get guide** – запрос текущего состояния автогида. Возможные ответы: `ON` или `OFF`.

– **get scope** – запрос текущего состояния телескопа. Возможные состояния: `PARK`, `SLEW`, `TRACKING`, `STAND`, `SEARCHING` или `CENTERING`.
– **get offset** – запрос смещения гидировочной звезды относительно оптической оси искателя (в горизонтальной системе в секундах дуги).

Координаты точки наведения вводятся в строковом виде. Например: `20 run ra=''20 46 13'' dec=''+33 58 13''` или `20 run az=''123 23 07'' alt=''+55 44 12''`. Часовое ведение автоматически включается после наведения в экваториальной системе координат и выключается после наведения в горизонтальной системе.

При инициализации телескопа включается питание телескопа, устанавливается время, дата и часовой пояс. При этом предполагается, что телескоп находится в заданном положении с координатами `az=''000 00 00''` и `alt=''+00 00 00''` (парковочное положение). Если телескоп при включении находится в другом положении, то его система координат окажется неправильной. При парковке телескоп устанавливается в парковочное положение и питание телескопа выключается.

Если выполняется гидирование по каналу DIMM, то одновременно определяется положение звезды и в камере искателя. Получающееся смещение между оптическими осями искателя и прибора позволяет автоматически корректировать положения оптической оси искателя в течение работы. Для того, чтобы при следующей инициализации программы оптические оси оказались совмещенными необходимо вручную обновить параметры `Operations/Centering/OpticalCenter` в конфигурационном файле `tlsp.cfg`.

9 Программа dome

Программа предназначена для управления укрытием, питанием машины druid и прибора MASS/DIMM и освещением подкуполного пространства. Команды приведены в списке:

– **run dome=open** и **run dome=close** – открытие и закрытие купола АСМ.
– **set limit=...** – установка предела открытия купола. Полностью открытый купол $\approx$ 700, а закрытый — 0.
– **set power=on** и **set power=off** – включение и выключения питания машины `druid` и прибора MASS/DIMM.
– **set light=on** и **set light=off** – включение и выключение светодиода освещения подкуполного пространства.
– **get dome** – запрос состояния купола. Возможные значения: `OPENED`, `CLOSED` и `UNDEFINED`.
– **get position** – запрос текущего положения створки купола в условных единицах (см. команду **set limit=...**)
– **get power** – запрос состяния питания аппартуры. Возможные ответы: `ON` и `OFF`.
– **get light** – запрос состяния освещения купола. Возможные ответы: `ON` и `OFF`.

10 Программа monitor

Постоянно запущенный демон для сбора информации об окружающих условиях с трех датчиков температуры, датчика влажности и анемометра. Дополнительно отслеживается

значение напряжения на аккумуляторах и в приборном ящике под куполом. Также фиксируется состояния двух концевиков, один из которых установлен на входной двери вагона, а второй — резервный. Первоначально регистрировалась скорость вращения ветрогенератора, однако этот датчик достаточно быстро вышел из строя и более не использовался.

Эта программа позволяет только запросить текущие данные об окружающих условиях, других функций и команд нет.

- **get meteodata** – запрос основных метеопараметров. Возвращаются минутные средние измеряемой на мачте внешней температуры `TEMP_EX`, температуры в вагоне `TEMP_IN`, средней скорости ветра `WIND` в м/с, максимальной скорости ветра `WIND_MAX`, направления ветра `WIND_DIR`, влажности воздуха `RH` и вычисляемой температуры точки росы `DEW_P`.
- **get support** – запрос дополнительных параметров, относящихся к жизнеобеспечению АСМ. Выдаются скорость вращения генератора `GEN_RPS`, напряжение на аккумуляторах `V_ACC`, температура в корпусе контроллера актюаторов `TEMP_BOX` и напряжение на контроллере `V_BOX`, состояние выключателей `SW1` и `SW2`.
- **get wind** – запрос характеристик ветра.
- **get power** – запрос напряжений питания аппартуры `V_ACC` и `V_BOX`.

Программа записывает данные измерений в выходной файл каждые 2 с и минутные средние значения. Пример строки средних:

```
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
```

Значения полей следующие:

1. M-префикс.
2. Дата в формате YYYY-MM-DD.
3. Момент окончания измерения, приводится UT в виде hh:mm:ss.
4. Внешняя температура (на мачте) в °C.
5. Внутренняя темепратура (в вагоне) в °C.
6. Средняя скорость ветра в м/с.
7. Максимальная за минуту скорость ветра
8. Азимут направления скорости ветра в градусах (отсчитывается от севера к востоку).
9. Влажность в процентах.
10. Поле, зарезервированное для давления.
11. Температура точки росы в °C.
12. Скорость вращения генератора.
13. Напряжение на аккумуляторах (Вольт).
14. Температура в корпусе контроллера актюаторов на вышке в °C.
15. Напряжение на вышке (на контроллере актюаторов) (Вольт).
16. Состояние концевика 1 двери вагона (`on` – дверь закрыта).
17. Состояние концевика 2

Строка мгновенных значений (префикс m) не содержит полей `WIND_MAX` и `DEW_P`, поскольку эти значения вычисляются на этапе усреднения данных измерений.

11 Программная организация автоматической работы

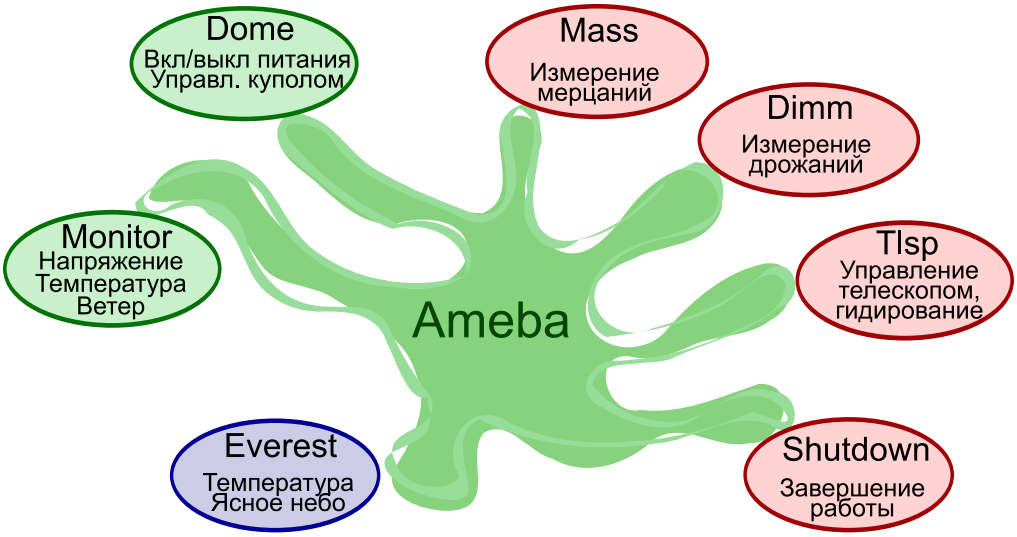

Рис. 4: Взаимодействие и структура программного обеспечения ACM при работе в автоматическом режиме

Общей организацией измерений занимается программа `ameba`, работающая на постоянно включенной машине `eagle`. С точки зрения межпрограммного обмена она выполняет роль клиентов для остальных программных компонентов. Для упрощения логической структуры, хотя это приводит к некоторым потерям времени, программа работает в одном потоке и практически все операции осуществляются последовательно.

Программа запускается каждый вечер демоном `cron` между 17 и 19 часами местного времени в зависимости от сезона. Несколько ранее на машине `omicron` запускается программа `dome` чтобы к началу наблюдений обеспечить включение аппаратуры. При старте `ameba` вычисляет моменты начала и конца ночи при заданном условии: высота Солнца равна $-9°$ и до начала ночи находится в режиме ожидания, периодически опрашивая программы метеослужбы `monitor` и `everest`.

Для начала измерений атмосферной турбулентности должны быть выполнены следующие условия:

1. Наступила ночь по установленному критерию.
2. Данные с датчика неба показывают, что небо ясное *(T_{sky} ниже некоторого предела).*
3. Скорость ветра не превышает установленный предел.
4. Напряжение на аккумуляторах выше минимально допустимого.

Планировалось включить в список условий и результаты измерений относительной влажности, но оказалось, что используемый датчик при большой влажности сам запотевает и начинает выдавать сильно заниженные значения.

Если условия выполнены, то начинается сессия измерений: совокупность процедур подготовки, проведения и завершения измерений атмосферной турбулентности. Каждая сессия включает следующую последовательность действий:

1. Открывание купола, включение машины `druid` (при ее включении автоматически запускаются программы `mass`, `dimm` и `tlsp`), инициализация программных компонентов.
2. Выбор объекта по условиям блеска, минимальной и максимальной высоте над горизонтом, расстоянию от Луны, доступной продолжительности измерений.
3. Наведение на объект, поиск объекта, его центрирование, верификация по соответствию измеренного потока ожидаемому, запуск процесса гидирования.
4. Измерение объекта в течение 1.5 часов или до нарушения одного из условий выбора.
5. Измерение фона до и после начала измерений выбранной звезды, но не реже 1 раза в час.
6. При неудачном наведении (не найден объект, его поток не соответствует ожидаемому, большой фон) — перевод на следующую программную звезду.
7. При ошибке возникшей в процессе измерений (поток от объекта значительно уменьшился, сбои в работе аппаратуры) — перевод на следующую программную звезду.
8. При появлении неустранимой ошибочной ситуации (ошибка возникает 3 раза подряд) или нарушении условий наблюдений происходит завершение сессии: парковка компонентов и аппаратуры, закрывание купола, выключение машины `druid`.

После завершения сессии по ухудшению внешних условий `ameba` ожидает восстановление условий для наблюдений. Чтобы предотвратить случайные эффекты, в систему детектирования условий введен значительный гистерезис. Так, если сессия наблюдений завершится по уловию "скорость ветра больше 9 м/с", то возобновится работа только тогда, когда в течении ≈30 мин. скорость ветра не будет превышать 6 м/с. Аналогично, если температура неба T_{sky} превысила $-21°$C, то наблюдения завершаются и будут возобновлены при условии, что в течении 20 – 30 минут $T_{sky} < -22°$C.

После завершения сессии по ошибочной ситуации (ошибка наведения или измерения, отсутствие подходящего объекта для измерений, слишком большой фон и т.п.) следует пауза в 20 – 30 мин.

Окончательное завершение наблюдений происходит в расчетный момент окончания ночи. Программа последовательно завершает текущую сессию наблюдений, отсоединяется от метеослужб `monitor` и `everest` и завершает свою работу. Если же АСМ находится в режиме ожидания, то завершение происходит несколько раньше, за 20 – 30 мин до окончания ночи, поскольку начинать новую сессию уже бессмысленно.

Точные значения упомянутых пауз и пределов для определения условий наблюдений содержатся в конфигурационном файле и уточняются по мере накопления опыта. На Рис. 5 показана логическая схема функционирования программы `ameba`, слева: условия начала сессии наблюдений, справа: процесс выполнения измерений.

В январе 2009 г. в программу были добавлены процедуры, обеспечивающие фотометрические наблюдения с целью точного определения коэффициентов экстинции в полосах чувствительности приборов MASS и DIMM. Для этого при переводе на следующий объект основной программы из специального списка выбираются 2 звезды для равновысотной фотометрической привязки и одна для определения экстинкции.

Для измерений турбулентности используется обновленный список `mass_star.lst`[7], содержащий 119 ярких звезд. Из него же программа выбирает объекты для подпрограммы фотометрических измерений по следующим условиям: звезда не должна быть переменной,

[7] см. `http://dragon/mass/download/doc/new_mass_cat.pdf`

должна иметь показатель цвета $B-V$ не больше $0^m.4$ и должна иметь склонение в интервале $90-\phi \pm 30$.

12 Вспомогательные программные средства

12.1 Демон shutdown

Эта вспомогательная программа–демон, запускаемая на машине `druid` при старте, решает одну единственную задачу: выключение этой машины по команде через TCP/IP соединение. Для предотвращения ситуации, когда один пользователь (программа `ameba`, например) может прекратить работу машины невзирая на остальных пользователей, демон `shutdown` обрабатывает блокировочные запросы пользователей. После того, как необходимость в работе машины у данного пользователя отпала, он должен снять свою блокировку. Программа не нуждается в инициализации и парковке и обрабатывает три команды:

- **run shutdown** – команда выключения машины. Выполняется только при отсутствии блокировок.
- **set lock=.....** – установка блокировки выключения. Ответ на команду показывает текущее число установленных блокировок после выполнения команды. Аргументом является символьная строка любой длины: ключ блокировки.
- **set unlock=.....** – снятие блокировки выключения. Ответ на команду показывает текущее число оставшихся после выполнения команды блокировок. Для успешного выполнения команды аргумент должен совпадать с ключом блокировки.
- **get lock** – запрос количества установленных блокировок.

Для того, чтобы программа `shutdown` могла выключить машину, сохраняя при этом локальную безопасность, используется специальная настройка утилиты `sudo`, позволяющая пользователю `mass` запускать системную утилиту `/sbin/shutdown`.

12.2 Синхронизация времени

Синхронизация времени на всех машинах осуществляется стандартным способом с с помощью постоянно запущенного демона `ntpd` по протоколу `ntp`. Используются следующие узлы `ntp` сервиса:

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

При реальных задержках канала связи (порядка 700 мс) время на машинах АСМ известно, как правило, с точностью лучше $0.02 \div 0.03$ с.

12.3 WEB–камеры

Две обзорных WEB-камеры (`dome` и `yard`), подключенных к машине `omicron`, обслуживаются программой `camsource 0.7.0` (`http://camsource.sourceforge.net`) представляющей собой демон захвата видеоизображений и простейший http–сервер. Хотя эта программа не

поддерживается уже несколько лет и содержит определенные ошибки, как демон она работает устойчиво, загрузка машины при этом минимальна. Ширины пропускания канала usb-контроллера машины `omicron` не хватает для полноценной работы с двумя камерами, поэтому драйвер Philips камер `pwc` запускается в режиме максимальной компрессии.

CGI-скрипт `eremaea`, запускаемый на машине `eagle` раз в минуту, запрашивает по протоколу http кадры с обеих камер и сохраняет их в соответствующем архиве. Срок хранения кадров с камер составляет 5 дней. По истечении срока хранения файлы изображений удаляются демоном. Доступ к изображениям возможен по адресу `http://eagle/eremaea`.

12.4 Архивация данных

В силу разных причин, стандартная утилита `logrotate` не подходит для автоматической архивации выходных файлов программных компонентов ACM. Эта задача решается при помощи скриптов `rotate.sh`, размещенных на каждой машине комплекса, и CGI-скрипта `pimenta`, функционирующего на машине хранения архивов `eagle`. Запускаемый раз в сутки демоном `cron` скрипт `rotate.sh` находит среди выходных файлов те, которые не изменялись больше суток, сжимает их компрессором `bz2` и с помощью утилиты `cUrl` отправляет методом PUT протокола http на сервер `eagle`, где `pimenta` размещает эти файлы в соответствующих директориях.

12.5 WEB–сайт http://eagle.sai.msu.ru

Для публичного доступа к информации о текущих условиях на ACM разработан WEB-сайт. Программа генерация страниц сайта написана на C++ (пакет www_eagle) и работает через интерфейс FastCGI — клиент-серверный протокол взаимодействия веб-сервера и приложения. Для соответствующей настройки http–сервера Apache в пакете имеются конфигурационные файлы. Необходимые для отображения данные собирает специальный демон `sensord`, использующий для этого протокол SNMP (Simple Network Management Protocol). Данные поставляют следующие сетевые узлы:

- две точки доступа (данные о входном/выходном трафике)
- omicron (запрашиваемые у программы `monitor` данные)
- eagle (данные с датчика ясного неба, запрашиваемые у программы `everest`)

Конвертация данных в SNMP осуществляется специальными бинарными динамическими модулями для `net-snmpd`: `rainAgent` и `meteoAgent`. Для построения графиков используется широкоизвестный пакет утилит RRD (Round Roubin Databases)[8], базы данных которого пополняет `sensord`.

На Рис. 6 показан пример вкладки ‘‘`Weather`’’ WEB-сайта. В интервале от 21 до 24 часов среды нижняя кривая на графике температур иллюстрирует описанный в тексте эффект “запотевания” датчика влажности, приводящий к нереалистично низким значениям RH и Dewpoint.

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

13 Приложение A

Таблица 1: Справочные данные о программных компонентах

Компонент	Машина	Внутренний IP	Порт	CVS модуль
`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

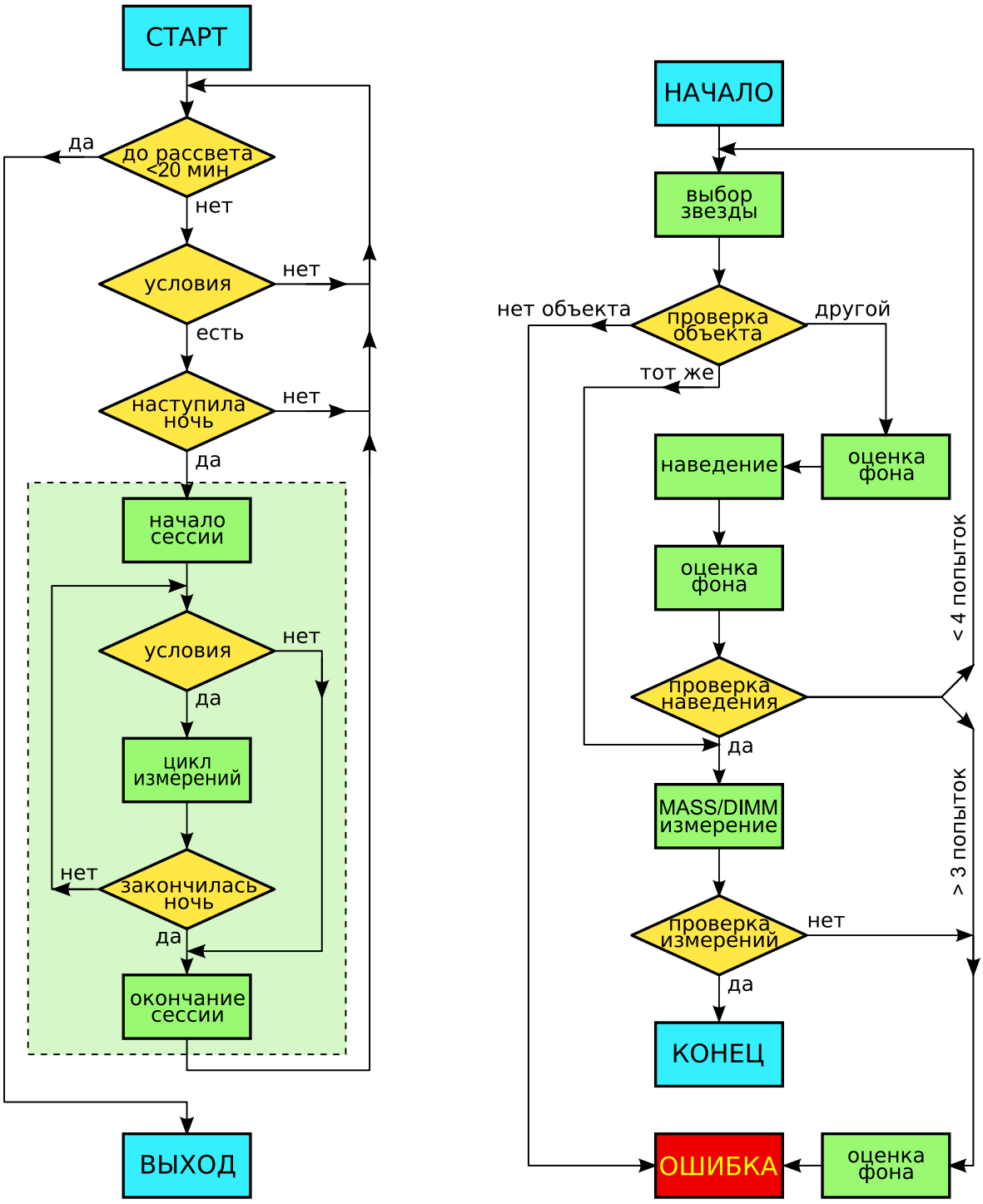

Рис. 5: Структурная схема функционирования программы `ameba`

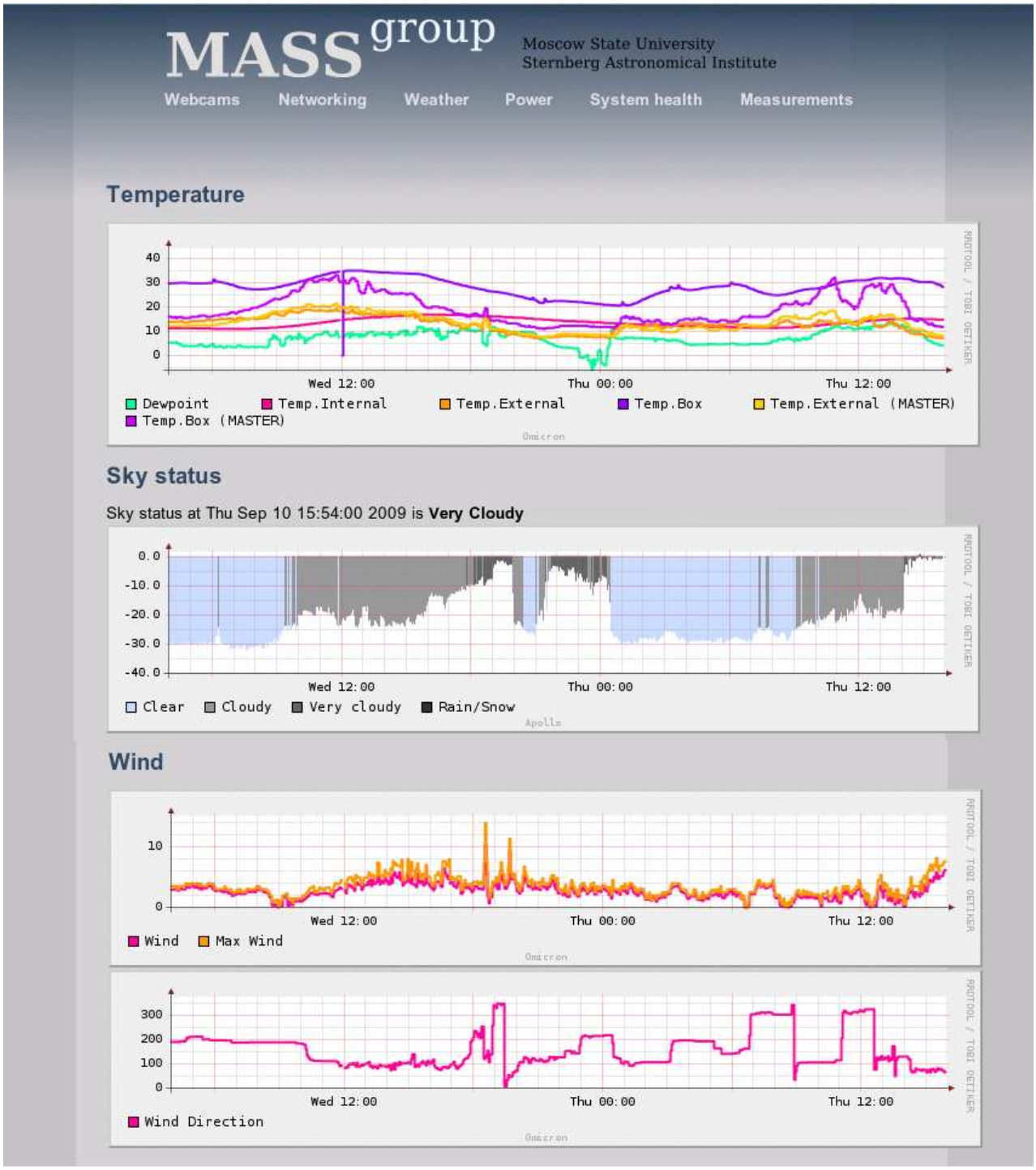

Рис. 6: Вид страницы "`Weather`" WEB-сайта `http://eagle.sai.msu.ru`. Сверху вниз: 6 температурных зависимостей, измеряемых в разных точках; поведение температуры неба: серый цвет – облачно, голубой – ясно; скорость и направление ветра.