Система температурного мониторинга главного зеркала БТА

Емельянов Э.В.

2024-09-09

Содержание

Введение **2**

1 Топология системы **2**

2 Протоколы связи **6**

2.1 CAN-протокол 6

2.2 Текстовый протокол данных при работе через USB 9

3 Основные утилиты и демоны **11**

Введение

В техническом отчете № 338 за 2018 г. описана первоначальная система температурного мониторинга главного зеркала (ГЗ) БТА, установленного на «переполированное» ГЗ № 1. В 2019 г. при обратной замене уничтоженного ЛЗОС ГЗ № 1 на ГЗ № 2 система была перенесена, но уже с увеличением количества термодатчиков: 20 датчиков установлено на тыльной части зеркала и 80 — на «донышках» углублений под разгрузки (т.е. фактически, на расстоянии около 20 см от рабочей поверхности ГЗ). Это позволило строить значительно более детализованные картины распределения температур. Также были учтены проблемы с использованием для питания контроллеров шести свободных проводников витой пары, оставшиеся два из которой использовались для организации связи по CAN шине. Внутри оправы был проложен специальный кабель для камер видеонаблюдения, включающий в себя витую пару и кабель питания (к сожалению, при монтаже выяснилось, что кабель — не медный, а изготовлен из «обмедненного алюминия», что делает его чрезвычайно хрупким). Разъемы RJ-45 на контроллерах были заменены на DB-9, имеющие значительно более надежный контакт, кроме того, адресация контроллера расширена до четырех бит, позволяя таким образом установить до 16 контроллеров (т.е. подключить до 256 термодатчиков в одну сеть).

Микропрограммное обеспечение контроллеров претерпело ряд изменений, последнее из которых было произведено в сентябре 2024 г (`build #62 @ 2024-09-04`). Один из контроллеров начал периодически блокировать CAN-шину постоянно повторяющимися «широковещательными» запросами на измерение температуры. Был немного изменен протокол, добавлено несколько дополнительных команд, а также запрещены широковещательные запросы. Кроме того, была удалена поддержка приема команд по UART, т.к. слишком низкая скорость этого интерфейса требует выделения достаточно большого объема памяти для текстового буфера, иначе часть сообщений из CAN-шины микроконтроллер просто не успевает зафиксировать (при получении ответа от очередного контроллера со свежими данными температур, особо остро эта проблема проявлялась при получении ответа от всех контроллеров на «широковещательный» запрос).

В данном документе описано состояние системы на сентябрь 2024 г. Опущены все факты, упомянутые в вышеназванном техническом отчете (его, как и данный документ, можно найти в репозитории с кодом [1]).

1 Топология системы

Основной компьютер, осуществляющий доступ к свежесобранным данным, установлен в металлическом щитке на правой стене помещения третьего этажа фокуса «H2». Помимо компьютера в щите установлен блок питания 12 В для энергоснабжения контроллеров, а также контроллер с номером «0», выполняющий роль ведущего для сбора данных с пяти контроллеров в оправе ГЗ БТА. В целях безопасности контроллер подключен к компьютеру по USB через гальваноразвязку. Питание ведомых контроллеров заводится через нормально замкнутое USB-реле, также подключенное к компьютеру.

Кабель (двойная витая пара) от ведущего контроллера внутрь оправы проложен еще в 2018 г., и из-за сложности прокладки при реорганизации системы в 2019 г не заменялся, поэтому питание передается в оправу по семи парам (по семь проводников на каждый полюс), восьмая пара используется в коммуникационных целях.

[1] `https://github.com/eddyem/tsys01`

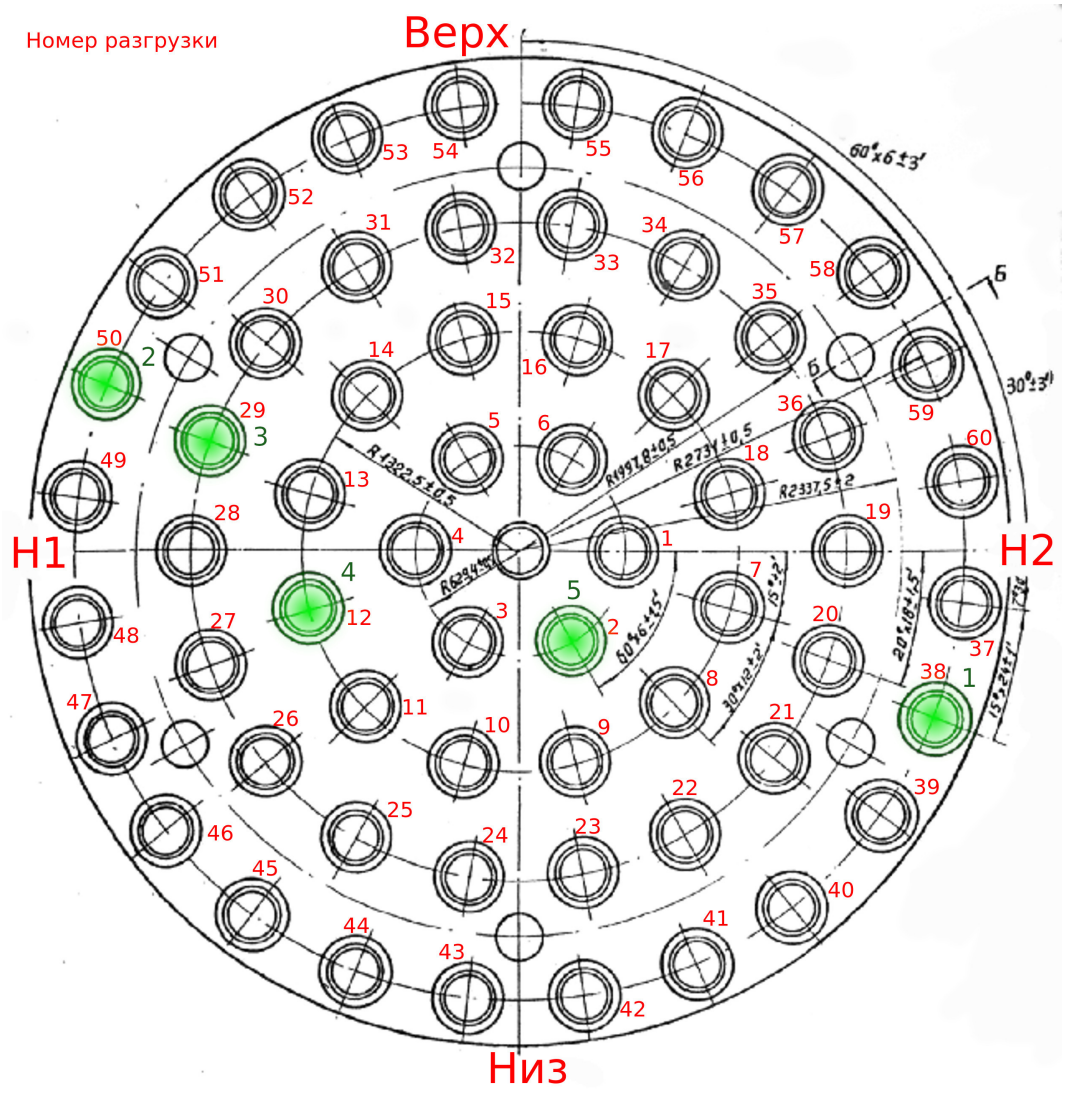

Рис. 1: Нумерация разгрузок ГЗ БТА. Зелеными пятнами отмечено расположение ведомых контроллеров, зеленая цифра — номер контроллера.

В оправе ГЗ (крышка вблизи Н2) расположен коммутационный разъем, соединяющий кабель со стороны ведущего контроллера и кабели CAN-шины, идущие к ведомым контроллерам, а также к частотному преобразователю, управляющему перемешиванием воздуха внутри оправы. К ведомым контроллерам проходит двойной кабель: витая пара (используется только одна пара) и питание. В предыдущем варианте питание передавалось по трем парам одинарной витой, что вызывало значительную просадку напряжения: на последнем контроллере питающее напряжение составляло всего лишь 6. . . 7 В.

На рис. 1 отображена нумерация разгрузок ГЗ БТА. Зеркало изображено так, как оно видится из первичного фокуса (в дальнейшем все координаты и изображения приведены к этому положению).

Ведомые контроллеры расположены на стенках оправы ГЗ БТА со стороны следующих разгрузок (номер контроллера – номер разгрузки): 1 – 38, 2 – 50, 3 – 29, 4 – 12, 5 – 2. Кабель проложен в окнах между разгрузками, за исключением тех мест, где окон не было: в них он проходит через специально подготовленное отверстие.

По окнам в оправе расположены кабели, подключающие термодатчики к контроллерам.

Условная нумерация термодатчиков состоит из трех цифр: первая цифра – номер контроллера (1–5), вторая – номер канала (0–7), третья – номер датчика в канале (0, 1). Таким образом, например, номер 471 соответствует датчику номер 1 в седьмом канале четвертого контроллера.

К термодатчикам припаяны экранированные четырехканальные кабели (каждый проводник в собственном экране). Каждая пара кабелей сводится в одной точке и переходит в витую пару, которая уже заканчивается непосредственно разъемом 4P4C, подключаемым к соответствующему гнезду на плате контроллера. Кабели к датчикам проходят через технологические отверстия в разгрузках ГЗ БТА. К сожалению, полная топология прокладки служебных кабелей не была изначально документирована и требует вскрытия заглушек всех разгрузок ГЗ БТА, что, ввиду незначительности задачи, видится бессмысленным.

Все термодатчики были аттестованы на массивной дюралюминиевой плите в диапазоне температур не хуже $-10 \div +20\,^{\circ}\mathrm{C}$, для каждого была построена кривая разницы показаний от медианной температуры группы опорных термодатчиков. Датчики, показывающие нелинейность свыше $\pm 0.1^{\circ}\mathrm{C}$ были отбракованы, для остальных был вычислен аддитивный показатель (погрешности ВЫЧИТАЮТСЯ из показаний датчика для получения реальной температуры). В таблице 1 приведено местоположение датчиков и вычисленные аддитивные поправки. Условные координаты — в дециметрах от центра зеркала, где направление вверх соответствует направлению к верху зеркала, а направление вправо — к H2. Местоположение (place) — номер разгрузки, а в случае датчиков, закрепленных на нижней поверхности зеркала — ближайшие разгрузки, между которыми наклеен датчик.

Датчики вблизи поверхности зеркала расположены на «донышке» каждого углубления под разгрузку. На нижней поверхности зеркала термодатчики установлены по шести радиусам: восемь на вертикальном диаметре (между каждой парой разгрузок около него) и по три на остальных радиусах с шагом 60°.

За время эксплуатации системы выяснилось также, что разъемы 4P4C, использующиеся для подключения датчиков, не отличаются особой надежностью. Из-за плохого контакта некоторые датчики могут перестать реагировать на опрос и исключаются (однако, контроллеры периодически проводят опрос всех датчиков, и если обнаруживается, что отключившийся датчик снова отвечает, производится переинициализация системы, и показания вновь становятся доступными).

Осмотр системы в 2024 г. показал, что на контроллере 1 деградировало несколько разъемов 4P4C. После замены контроллера основная часть датчиков, ранее не подававших признаков жизни, «ожила». Мертвым оказался лишь датчик 100, не реагирующий на опрос даже на самой медленной частоте шины I2C. Датчики 160 и 161 отсутствуют изначально (при монтаже в 2019 г. были уничтожены неопытным монтажником, не заземлившим паяльник). Учитывая незначительность потери, было принято решение так все и оставить, а не вскрывать разгрузки, чтобы заменить недостающие датчики. Кроме того, заменять-то их и нечем: среди оставшихся из 100 закупленных небольшая часть по тем или иным причинам неработоспособна, а оставшиеся имеют значительные (свыше $\pm 0.1^{\circ}\mathrm{C}$) отклонения по линейности. На контроллере 2 у датчиков 250 и 251 по причине плохого контакта изредка пропадают показания. Датчики 370 и 371 контроллера 3 вышли из строя (дают показания, но совершенно неверные). У контроллера 4 датчики 450 и 451 в основном не работают по причине плохого контакта. Датчик 551 контроллера 5 не подает признаков жизни.

Итого, из 80 закрепленных на ГЗ БТА датчиков надежные показания дают 70.

No	**Place**	ΔT	X	Y
100	36	-0.07	19	7
101	19	0.03	20	0
110	20	0.03	19	-7
111	20-21	-0.05	17	-10
120	40	0.02	17	-22
121	21	-0.03	15	-13
130	39	0.02	22	-17
131	38-39	-0.09	24	-14
140	38	0.03	25	-10
141	37	0.02	27	-4
150	60	-0.01	27	4
151	59	-0.09	25	10
160	58	0.07	22	17
161	58-59	0	24	14
170	56	0.02	10	25
171	57	0.08	17	22
200	55	-0.01	4	27
201	54-55	-0.03	0	27
210	54	0.04	-4	27
211	53	-0.01	-10	25
220	52	-0.02	-17	22
221	51	-0.04	-22	17
230	50	0.03	-25	10
231	50-51	-0.03	-24	14
240	48	-0.03	-27	-4
241	49	0.03	-27	4
250	47	-0.05	-25	-10
251	46-47	0	-24	-14
260	46	0.03	-22	-17
261	45	-0.05	-17	-22
270	44	-0.02	-10	-25
271	43	0.04	-4	-27
300	24	0.12	-3	-20
301	25	0	-10	-17
310	26	-0.08	-15	-13
311	26-27	-0.08	-17	-10
320	27	-0.1	-19	-7
321	28	-0.04	-20	0
330	29	-0.08	-19	7
331	29-30	0.03	-17	10

No	**Place**	ΔT	X	Y
340	30	0.11	-15	13
341	31	0.07	-10	17
350	32	0.09	-3	20
351	32-33	-0.09	0	20
360	33	-0.1	3	20
361	34	0.04	10	17
370	35	0.05	15	13
371	35-36	-0.04	17	10
400	17	0.01	9	9
401	17-18	-0.04	11	7
410	16	0.01	3	13
411	15-16	-0.04	0	13
420	14	0.14	-9	9
421	15	0	-3	13
430	13	0.07	-13	3
431	13-14	-0.02	-11	7
440	12	0.04	-13	-3
441	11-12	-0.06	-11	-7
450	11	0.01	-9	-9
451	10	0.08	-3	-13
460	9	-0.09	3	-13
461	9-10	-0.04	0	-13
470	23	0.06	3	-20
471	23-24	-0.05	0	-20
500	42	0.05	4	-27
501	42-43	-0.08	0	-27
510	22	0.1	10	-17
511	41	-0.05	10	-25
520	8	0	9	-9
521	7-8	0.01	11	-7
530	2	0	3	-5
531	2-3	-0.02	0	-6
540	3	0.05	-3	-5
541	4	-0.03	-6	0
550	5	0.03	-3	5
551	5-6	0.06	0	6
560	6	0	3	5
561	1	-0.06	6	0
570	7	-0.07	13	-3
571	18	0.02	13	3

Таблица 1: Положение термодатчиков на ГЗ БТА.

2 Протоколы связи

Протокол связи с собственно термодатчиками по шине I2C документирован производителем, поэтому здесь не озвучивается.

Реализацию всех протоколов можно найти в исходных кодах по вышеупомянутой ссылке.

2.1 CAN-протокол

Данные по CAN-шине передаются пакетами переменной длины: от одного до восьми байт. Каждый контроллер (за исключением работающего в режиме мониторинга шины) принимает лишь пакеты с определенным идентификатором, равным $0x680 + N$, где N – номер контроллера, устанавливаемый перемычками на плате. Контроллеры первого поколения имели лишь три бита адресации, в следующем поколении адресация расширена до четырех бит, однако, протокол все еще рассчитан на адреса от 0 до 7, где 0 — ведущий (автоматически запускается в режиме мониторинга, если не получит команду перейти в режим индивидуальной адресации), а остальные — ведомые (аналогично, могут быть переключены в режим мониторинга командой по USB). «Широковещательные» сообщения, поддерживаемые первым поколением контроллеров, удалены с целью защиты шины от «спамеров». Как и удалена поддержка бесполезного ввиду слишком медленной скорости UART.

В каждом пакете нулевым байтом является маркер типа пакета (`MAR`): данные (`0x5A`) или команда (`0xA5`). Байт 1 (`NUM`) — номер контроллера, отправившего сообщение (команду или данные), что позволяет выявить адресата и отправить ответ именно ему (таким образом, ведомые контроллеры тоже могут посылать команды ведущему). Байт 2 (`CMD`) — код команды (перечислены далее). Оставшиеся байты (`DATA`) содержат данные ответа на запрос и присутствуют лишь в пакетах с маркером «данные». Таким образом, графически формат пакета можно выразить так:

Перечень команд (числовое значение команды, мнемоническое обозначение, расшифровка):

0x00 `CMD_PING` ожидание пакета данных с эхом этой команды (используется для проверки существования контроллера на шине);

0x01 `CMD_START_MEASUREMENT` начать одиночное измерение температуры (см. далее);

0x02 `CMD_SENSORS_STATE` получить состояние датчиков (см. далее);

0x03 `CMD_START_SCAN` войти в режим непрерывного измерения температуры (каждые 15 секунд контроллер отправляет данные с термодатчиков запрашиваемому узлу);

0x04 `CMD_STOP_SCAN` выйти из предыдущего режима;

0x05 `CMD_SENSORS_OFF` отключить питание датчиков;

0x06 `CMD_LOWEST_SPEED` минимальная скорость I2C (5.8 кГц);

0x07 `CMD_LOW_SPEED` низкая скорость I2C (10 кГц);

0x08 `CMD_HIGH_SPEED` высокая скорость I2C (100 кГц);

0x09 `CMD_REINIT_I2C` реинициализация шины I2C;

0x0A `CMD_CHANGE_MASTER_B` (исключена);

0x0B `CMD_CHANGE_MASTER` (исключена);

0x0C `CMD_GETMCUTEMP` температура микроконтроллера (см. далее);

0x0D `CMD_GETUIVAL` значения напряжений и токов (см. далее);

0x0E `CMD_GETUIVAL0` (см. далее);

0x0F `CMD_GETUIVAL1` (см. далее);

0x10 `CMD_REINIT_SENSORS` запуск процедуры обнаружения термодатчиков и получения их калибровочных величин;

0x11 `CMD_GETBUILDNO` номер версии сборки текущей микропрограммы;

0x12 `CMD_SYSTIME` условное (в миллисекундах) время, начиная с последнего старта микроконтроллера (данные начинаются с байта 4, остроконечный формат);

0x13 `CMD_USBSTATUS` состояние активности подключения к контроллеру по USB (в ответе 1, если USB активно);

0x14 `CMD_SHUTUP` после получения этой команды контроллер входит в режим молчания, не посылая никаких сообщений в CAN-шину;

0x15 `CMD_SPEAK` нормальный режим (контроллер отвечает в CAN-шину);

0xAA `CMD_ANSOK` ответ «OK» контроллера в случае, если не предусмотрена пересылка данных на запрос;

0xDA `CMD_DUMMY0` тестовая команда;

0xAD `CMD_DUMMY1` тестовая команда.

Ответ на команду `CMD_START_MEASUREMENT`, где `SNO` – номер датчика ($10 \cdot N + M$, где N – номер канала, M – номер датчика в паре), `TH` – старший байт данных и `TL` – младший байт данных (температура передается в сотых градуса Цельсия):

0	1	2	3	4	5
0x5A	NUM	0x01	SNO	TH	TL

При считывании температурных данных в случае ошибки измерений (значение температуры, полученное с шины, лежит вне допустимого диапазона) возвращается число -30000 (т.е. -300°C), а при ошибке передачи команды считывания измеренных данных — число -31000 (т.е. -310°C).

Ответ на команду `CMD_SENSORS_STATE`:

0	1	2	3	4	5	6	7
0x5A	NUM	0x02	ST	SP[0]	SP[1]	NS	NT

Здесь поле `ST` – состояние датчиков:

0x00 `SENS_INITING` питание включено, происходит инициализация;

0x01 `SENS_RESETING` процедура обнаружения датчиков командой сброса;

0x02 `SENS_GET_COEFFS` получение калибровочных коэффициентов обнаруженных датчиков;

0x03 `SENS_SLEEPING` «спящее» состояние между измерениями;

0x04 `SENS_START_MSRMNT` команда начала измерения температуры;

0x05 `SENS_WAITING` ожидание окончания измерений температуры;

0x06 `SENS_GATHERING` сбор температур с обнаруженных термодатчиков;

0x07 `SENS_OFF` питание датчиков отключено;

0x08 `SENS_OVERCURNT` превышение допустимого тока питания датчиков (0.5 А), датчики временно отключены;

0x09 `SENS_OVERCURNT_OFF` попытка включения питания датчиков более 32 раз закончилась состоянием `SENS_OVERCURNT`, питание отключено вплоть до внешнего вмешательства.

Поля `SP[x]` – битовые, каждый бит означает, что датчик с данным индексом обнаружен (1) или нет (0). `SP[0]` – датчики с нулевым адресом в паре (младший бит – канал 0, старший – канал 7); `SP[1]` – датчики с индексом 1 в паре.

Поле `NS` – количество обнаруженных датчиков (откликнувшихся на команду инициализации и сообщивших калибровочные данные).

Поле `NT` – количество измеренных температур, которое в силу ошибок может быть меньше `NS`, в этом случае рекомендуется снизить скорость I2C данного контроллера. Несмотря на то, что кабельное хозяйство расположено в эдакой «клетке Фарадея», из-за значительной (местами до 2.5 м) длины линии I2C рекомендуется первой же командой, переданной ведущему контроллеру, установить на всех ведомых минимальную скорость I2C.

Ответ на команду `CMD_GETMCUTEMP` (тупоконечный формат в сотых градуса Цельсия):

0	1	2	3	4
0x5A	NUM	0x0C	TH	TL

Ответ на команду `CMD_GETUIVAL` эквивалентен ответу на посылку последовательно команд с индексами 0 и 1.

`CMD_GETUIVAL0` – напряжение на шинах 5 В и 12 В (в сотых Вольты). Тупоконечный формат, байты 3 и 4 – 12 В, байты 5 и 6 – 5 В:

0	1	2	3	4	5	6
0x5A	NUM	0x0E	V12H	V12L	V5H	V5L

`CMD_GETUIVAL1` – потребляемый ток на шине 12 В (в миллиамперах) и напряжение питания МК (в сотых Вольты). Тупоконечный формат, байты 3 и 4 – ток, байты 5 и 6 – напряжение 3.3 В (закупленная на алиэкспрессе партия токовых датчиков полностью оказалась бракованной, поэтому ни на одном контроллере холловские датчики тока не распаяны):

0	1	2	3	4	5	6
0x5A	NUM	0x0F	I12H	I12L	V3.3H	V3.3L

2.2 Текстовый протокол данных при работе через USB

Формат команды: `[<num>]<char><endline>`, где `<char>` — символ команды (большая или малая латинская буква), а `<endline>` — символ завершения строки (`\n`). Параметр `[<num>]` используется лишь для основной части команд в верхнем регистре, означая номер контроллера, которому будет осуществлена передача следующей команды. Все команды в нижнем регистре запускаются исключительно локально на данном контроллере (существуют также команды, не имеющие «локального» варианта, как и команды, не имеющие «сетевого»). «Сетевые» команды могут быть отправлены с любого контроллера: как ведущего, так и ведомого. При получении команды контроллер сохраняет номер адресата, отправляя ответ с нужным ID.

Далее приводится список основных команд.

@ инвертировать флаг режима отладки: при установленном флаге контроллер выдает дополнительные сообщения в USB (на посылки в CAN-шину данный флаг не влияет);

A разрешение подчиненному узлу отправлять сообщения в CAN-шину (нормальный режим работы);

a получение «сырых» значений измерений АЦП микроконтроллера;

B посылка тестового сообщения по CAN-шине (на него не приходит ответа, лишь в USB получателя выводится сообщение «DUMMY», если, конечно, к USB кто-либо подключен);

b получение или установка (если за литерой следует число) скорости CAN-шины (в кбод)

c отобразить коэффициенты заводской калибровки для всех подключенных термодатчиков (сетевой вариант данной команды не предусмотрен: пока еще не было прецедентов, чтобы это понадобилось бы);

D послать по CAN-шине тестовое сообщение ведущему контроллеру (т.е. с идентификатором 0x580);

Ee выйти из режима сканирования температуры (в режиме сканирования каждые 15 с контроллер проводит измерение температуры с выдачей данных в CAN-шину с идентификатором «заказчика»);

Ff отключить питание термодатчиков (не рекомендуется часто отключать питание, иначе датчики быстро выйдут из строя);

g активировать режим «прослушивания» CAN-шины (автоматически включен у ведущего контроллера, но при подключении по USB к ведомому в диагностических целях может быть также активирован);

Hh переключить I2C в режим скорости 100 кГц (не рекомендуется для длинных линий);

Ii запустить процедуру поиска термодатчиков (необходимо в случае, если контроллер отключил их питание по аварийной ситуации);

Jj температура микроконтроллера (в сотых градуса Цельсия);

Kk получить все значения измеренных АЦП МК напряжений и токов (U3.3, U5, U12, I12 – см. выше);

Ll переключить I2C в режим скорости 10 кГц (режим по умолчанию, однако, как оказалось, линии длиной больше двух метров требуют перехода в «сверхмедленный» режим);

N получение значения номера сборки микрокода (для текущего контроллера при подключении по USB номер и дата сборки выводятся в шапке сообщения подсказки, например, при запросе «?»);

Oo (нестандартный: заглавная литера тоже означает локальную команду) включение (заглавная) или отключение (прописная) набортных диагностических диодов (светодиод LED0 пульсирует с частотой в 1 Гц, LED1 своим свечением диагностирует нормальную работу CAN-шины);

P ping — запрос на существование в сети контроллера с соответствующим адресом (ответом является отправка назад того же пакета данных, но с маркировкой «data»);

Qq получение условного (в миллисекундах) времени с включения контроллера: несмотря на значительную погрешность кварцевых генераторов, порядок величины позволяет определить, не перезагружался ли тот или иной контроллер после общего включения питания;

Rr переинициализация I2C;

S «заткнуть» данный узел: он перестанет отправлять в CAN-шину какие бы то ни было сообщения (вплоть до получения разрешения вернуться в нормальный режим), данная команда, отправленная ведомым узлом, может «заткнуть» и ведущий, что чревато;

s отправить в CAN-шину сообщение, первым байтом следует идентификатор сообщения, затем – до восьми байт данных (формат: двоичный с префиксом `0b`, восьмеричный с префиксом `0`, десятичный или же шестнадцатеричный с префиксом `0x`);

Tt запуск одиночного измерения температуры (стандартная команда для измерений по запросу с ведущего контроллера);

U состояние USB (в случае, если к контроллеру подключен хост, возвращается единица);

u режим приема сообщений лишь с идентификатором, равным $0x580 + ADDR$ (режим по умолчанию для ведомых);

Vv переключить I2C в режим наименьшей скорости (около 5.8 кГц);

Xx активировать режим сканирования температуры (ведомые каждые 15 с будут отправлять измеренные данные);

Yy состояние термодатчиков (см. выше);

z проверить флаг ошибки CAN-шины.

3 Основные утилиты и демоны

При подключении к компьютеру любого контроллера, udev-правило создает в директории `/dev` файл `tsyscontrX` (если подключено лишь одно устройство, `X` будет нулем), являющийся символической ссылкой на последовательный терминал, эмулятор PL2303 (udev-скрипт и создан в целях идентификации конкретного устройства, если к компьютеру подключено несколько разнотипных устройств, эмулирующих PL2303). Эмуляция именно PL2303 создана с двумя целями: избавиться от «перехвата» устройства демоном `modemmanager` systemd-based дистрибутивов, а также затруднить работу с устройствами из вражеской среды. Код данного скрипта:

```
ACTION=="add", DRIVERS=="usb", ENV{USB_IDS}="%s{idVendor}:%s{idProduct}"
ACTION=="add", ENV{USB_IDS}=="067b:2303", ATTRS{interface}=="?*",
    PROGRAM="/bin/bash -c \"ls /dev | grep $attr{interface} | wc -l \"",
    SYMLINK+="$attr{interface}%c", MODE="0666", GROUP="tty"
ACTION=="add", ENV{USB_IDS}=="0483:5740", ATTRS{interface}=="?*",
    PROGRAM="/bin/bash -c \"ls /dev | grep $attr{interface} | wc -l \"",
    SYMLINK+="$attr{interface}%c", MODE="0666", GROUP="tty"
```

Для возможности разделения последовательного устройства между несколькими независимыми процессами (опрос температурных данных, работа с вентиляторами, диагностика), запускается прокси-демон[2], предоставляющий сетевой доступ по заданному локальному INET-сокету или же UNIX-сокету. Полученные из последовательного устройства данные транслируются всем подключенным клиентам, а данные от клиентов записываются в устройство.

Основной сетевой демон[3] подключается к сокету прокси-демона, а также открывает общедоступный сетевой сокет, через который любой клиент может получить данные последних измерений. Кроме того, при помощи `gnuplot` каждые 15 минут формируются изображения с распределением температуры по обоим слоям установки термодатчиков. Данные доступны через веб.

При помощи `netcat` и баш-скриптов реализован также веб-интерфейс управления вентиляторами, расположенными внутри оправы ГЗ БТА. На рис. 2 продемонстрирован простейший интерфейс, доступный по адресу `http://mirtemp.sao.ru/fans.html`. Переключатели «Low», «Mid» и «High» позволяют установить заданную скорость вращения вентиляторов. Флажок «Run middle» включает вентилятор, расположенный в средней части зеркала (он захватывает воздух из подкупольного пространства, в отличие от остальных вентиляторов, которые лишь перемешивают внутри оправы ГЗ). При выборе флажка «Stop» выключаются все вентиляторы. Кнопка «Set» активирует текущий выбор. Под ней отображаются текущие значения потребляемого вентиляторами тока и скорость вращения их двигателей (центральный вентилятор включается пускателем, поэтому в статистике не участвует).

[2] `https://github.com/eddyem/eddys_snippets/tree/master/serialsock`

[3] `https://github.com/eddyem/tsys01/tree/master/src/netdaemon`

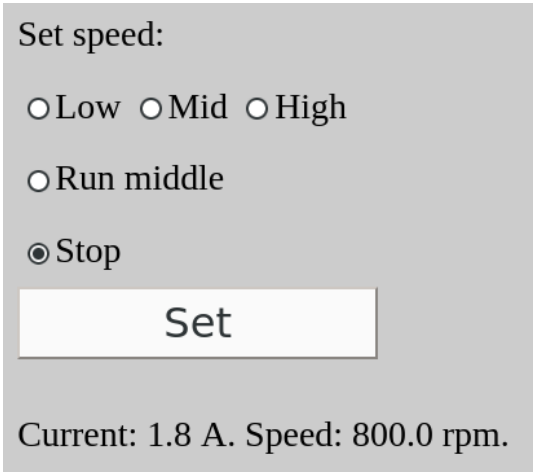

Рис. 2: Управление вентиляторами.

На рис. 3 и 4 приведены примеры изображений распределения температур. В тех местах, где термодатчики отсутствуют, либо же их показания выбиваются за 3σ от медианной по зеркалу, данные не отображены.

Вся подробная информация: исходные коды, схемы и т.п. размещена в репозитории[4].

Для подключения к сокету прокси-демона в отладочных целях удобно использовать терминал[5], позволяющий задавать формат входных и отображаемых данных, а также разделяющих строку ввода от поля вывода (в отличие от netcat, смешивающего все данные).

Для передачи данных в СУ БТА на основном компьютере управления (обычно, `acs7`) запущен демон `bta_mirtemp`. К сожалению, он «сляпан на скорую руку», поэтому даже не проверяет наличие уже запущенного экземпляра. В случае наличия данных от сервера `mirtemp`, в СУ вводятся полученные данные средней температуры по зеркалу (алгоритмы вычисления и т.п. — в репозитории). Если же в течение 15 минут данные отсутствуют, в системе отображается ошибка.

Разворачивание системы с нуля элементарно, и не требует каких-либо комментариев. Выдачей клиенту изображений распределений температур может заниматься NGINX или даже элементарный самописный сервер (не более трех-четырех страниц кода на C).

В будущем, если найдется разработчик, желающий довести систему до ума, он сможет сменить баш-скрипты управления вентиляторами на более защищенные демоны на C, а также внедрить систему интеллектуального контроля скоростью вращения вентиляторов (отключать их при открытых крышках и устанавливать скорость вращения в зависимости от градиента по зеркалу). Кроме того, уже который год мы планируем вместо центрального вентилятора установить управляемый кондиционер, который позволил бы поддерживать на заданном уровне температуру ГЗ БТА. Управление этим кондиционером — еще одна задача следующего поколения разработчиков.

[4] `https://github.com/eddyem/tsys01`
[5] `https://github.com/eddyem/tty_term`

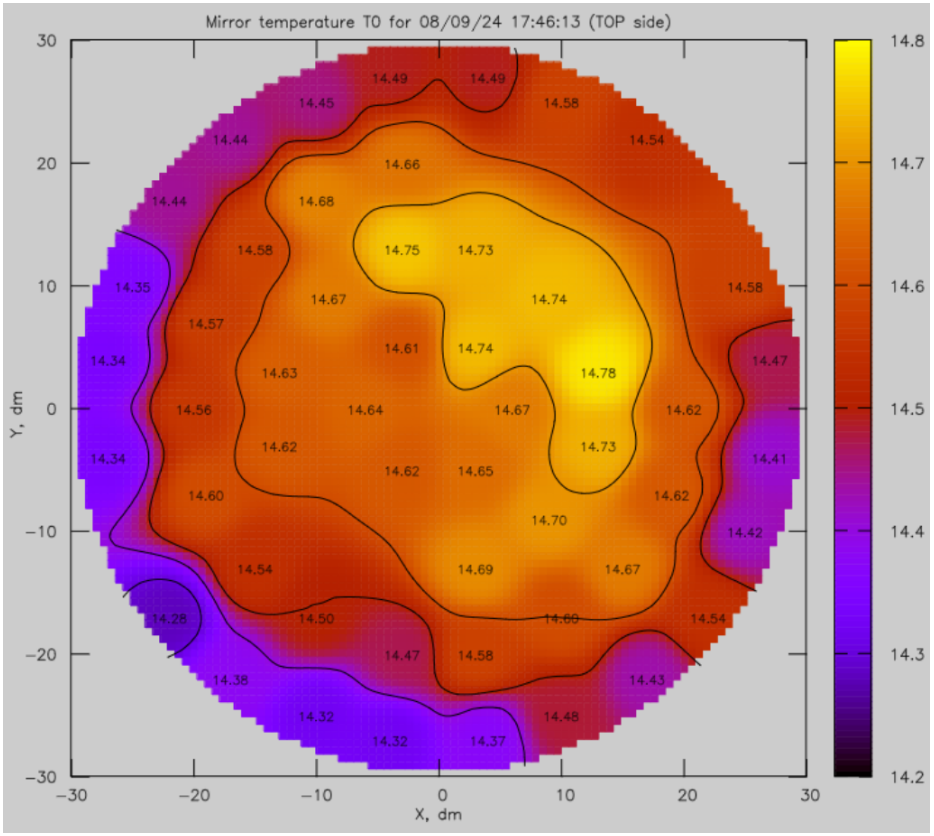

Рис. 3: Пример изображения распределения температур приповерхностного слоя.

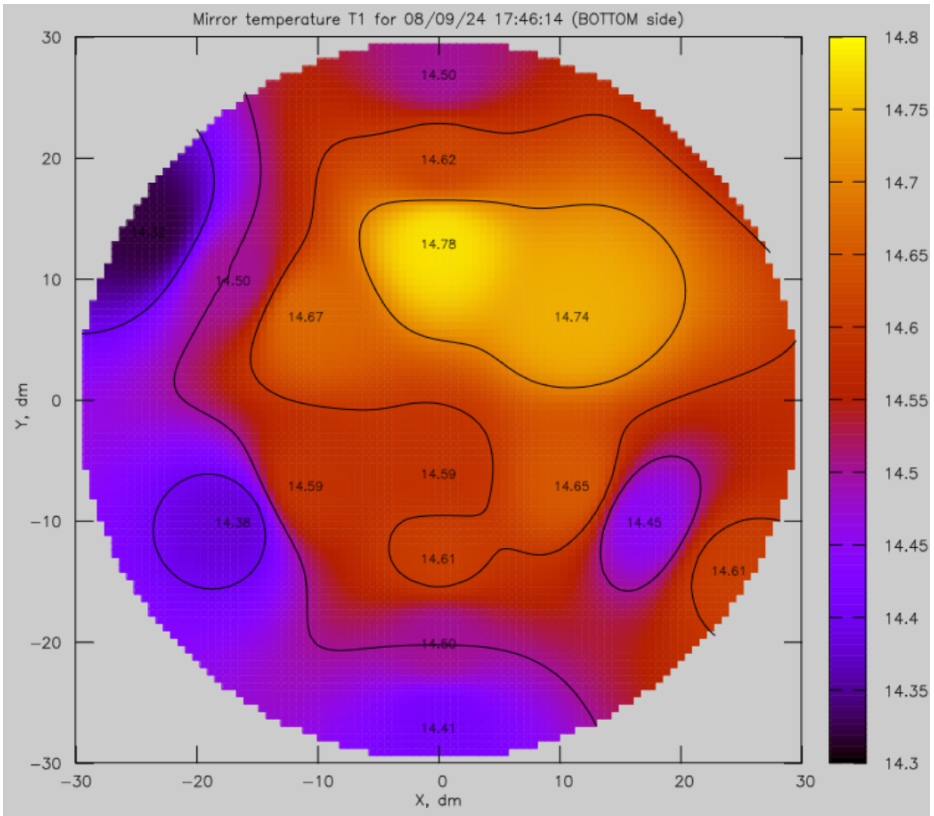

Рис. 4: Пример изображения распределения температур тыльной поверхности.