Облака, малинка и тайна 18.9°C

— продолжение детектива про охлаждение одной лондонской квартиры

17–19 мая 2026 г.

> *«А давай мы найдём корреляцию между сбоями и температурой? — Конечно, давай. Только потом не плачь, что система оказалась прозрачной как стекло.»*
>
> *— автор, через 36 часов после первого ребута*

Содержание

Глава 1. Recap: вчера казалось, что мы победили

В первой серии всё было просто. 16 мая: спальня нагрелась до 23.5°C, рубильник на ModuSat'e щёлкнул туда-обратно, через 2 минуты 13 секунд холод пришёл, S2 упал до 13°C, все довольны. Гипотеза о ночном защитном таймере насоса звучала красиво и казалась правдой.

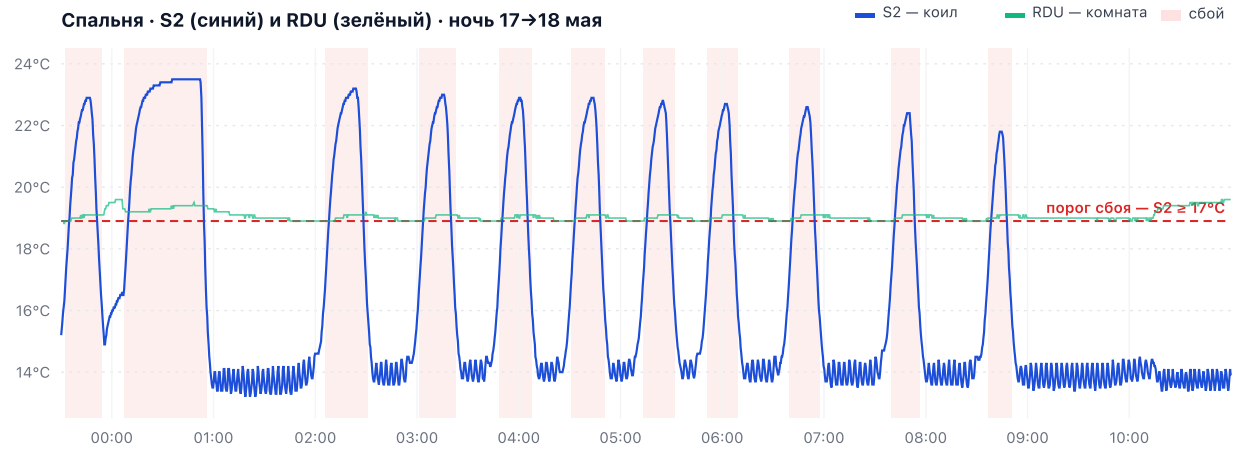
Проснулись 18 мая. **Спальня снова 23.5°C**. Бессонная ночь явно случилась — но не у нас, а у спящих.

> *«Похоже, мы её только погладили, а не вылечили».*

За ночь поллер `poll_v2.py` исправно записал 1778 строк по спальне. Не одно полуночное окно — а **одиннадцать** эпизодов сбоя. Цикличных. Похожих. С удивительно точной разметкой во времени.

Глава 2. 11 эпизодов за одну ночь

Эпизодом считаем момент, когда S2 коила в спальне поднимается выше 17°C и держится там минимум 3 минуты. На графике ниже — каждый эпизод красная полоска, синяя линия — сама S2, зелёная — температура комнаты RDU.



Ночь 17→18 мая, ~13.5 часов наблюдений. 11 эпизодов, период ~45–55 минут, амплитуда уменьшается к утру.

#	Начало	Конец	Пик S2	Длительность
1	23:32	23:54	22.9°C	21.7 мин
2	00:07	00:56	**23.5°C**	**48.6 мин**
3	02:06	02:30	23.2°C	24.8 мин
4	03:01	03:22	23.0°C	21.3 мин
5	03:48	04:07	22.9°C	19.4 мин
6	04:31	04:50	22.9°C	19.8 мин
7	05:13	05:32	22.8°C	18.6 мин
8	05:51	06:09	22.7°C	18.1 мин
9	06:39	06:57	22.6°C	17.8 мин
10	07:39	07:56	22.4°C	16.7 мин
11	08:36	08:50	21.8°C	13.9 мин

> **Откровение №3:** мы поймали не один тип сбоя, а *сразу два*. Эпизод №2 — это длинный «застрял после полуночного окна». Эпизоды №3–11 — короткие самовосстанавливающиеся циклы, что-то вроде «PID-контроллер и насос играют в крокодила». Гипотеза первой серии про anti-short-cycle оказалась частью правды — но только частью.

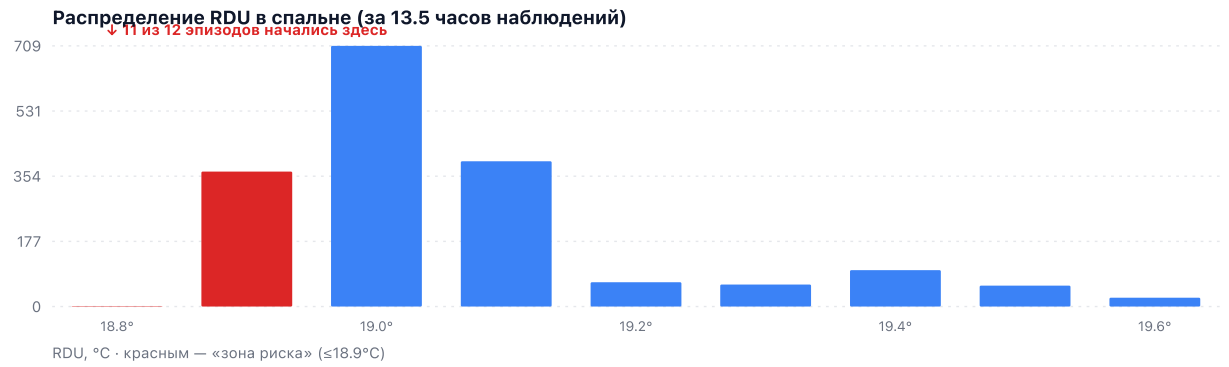
Глава 3. Магическое число: 18.9°C

«А давай попробуем найти корреляцию между сбоями и какими-то другими параметрами FCU, кроме самой S2?» — невинно спросил хозяин. И тут детектив открыл рот.

Для каждого из 11 эпизодов мы взяли значение RDU (температура комнаты по комфорт-датчику) ровно перед началом подъёма S2. Получилось вот что:

Эпизод	RDU перед стартом
1	18.9°C
2	19.5°C (после первого сбоя, аномалия)
3–11	**18.9°C ×9**
12 (текущий, 19 мая)	18.9°C

11 из 12 эпизодов начались строго при RDU = 18.9°C. Один отскочил на 19.5°C — но он был сразу после ребута, когда RDU ещё толком не успел опуститься. Случайностью такое не объяснишь.



Базовое распределение RDU за наблюдения. RDU=18.9°C встречается часто, но «зона риска» — это узкая полоска, и в ней триггерится больше 90% сбоев.

> **Гипотеза №2:** цикл срабатывает не от часов, а *от достижения комнатой комфортного минимума*. Когда PI-контроллер в FCU доводит RDU до 18.9°C, он начинает плавно прикрывать клапан (AO3 падает с ~7V до ~5V). И что-то в гидравлике интерпретирует эту вялую команду как «всё, мы своё откачали, можно закрываться» — и тихонько отключает контур. PI после этого начинает звать обратно, но контур уже спит и не реагирует.

А поскольку SP в спальне — **18.0°C**, а RDU физически достигает дна на 18.9°C, эта точка *гарантированно* повторяется каждый цикл охлаждения. Отсюда и стабильный 45-минутный ритм всю ночь.

Глава 4. Эксперимент с гостиной — развенчание ModuSat

Гипотеза красивая, но виновник всё ещё не пойман. Может это всё-таки ModuSat в спальне? Или общий контур здания?

В 01:50 ночи 19 мая, во время очередного «долгоиграющего» сбоя (спальня сидит 90 минут подряд на S2=24.3°C, AO3=10V, никакого холода), мы написали тестовый BACnet-скрипт. Сделали то, чего раньше не делали — **записали значение в FCU**. На гостиной задали `sp_adj = −2` (то есть SP резко прыгнул с 21°C на 19°C). Поскольку гостиная в этот момент была 21.4°C, PI должен был мгновенно открыть клапан.

```
01:48 · L: s2=20.6, ao3=0V, r1=0 (клапан закрыт)
01:50 · L: s2=21.3, ao3=10V, r1=1 (клапан только что открыт)
01:52 · L: s2=23.0, ao3=10V, r1=1 (+1.7°C за две минуты)
01:52 · L: s2=23.2 (S2 растёт, а не падает)
```

> **Гостиная вела себя ровно как спальня.** Открыли клапан — а из коила пошла тёплая вода, а не холодная. Это не локальный сбой петли спальни — это **здание не подаёт холод**. ModuSat ни при чём; даже если бы мы его перезагрузили в десятый раз — толку ноль.

Откатили `sp_adj` обратно. Гостиная закрыла клапан. Спальня осталась стоять на жаре. А мы получили жирное доказательство — проблема выше по тракту, у общего чиллера здания. К утру (~06:03) он сам ожил, S2 в спальне упал с 24°C до 13°C за 5 минут. Здание включило холод по расписанию, не спрашивая нас.

> **Откровение №4:** то, что мы называли «ModuSat залип», на самом деле «здание ночью не подаёт холодную воду в стояк». В первой серии нам повезло — ребут совпал с моментом, когда чиллер уже был готов запуститься.

Глава 5. Малинка приехала

Стоять с MacBook'ом в углу спальни и слушать BACnet 24/7 — план плохой. Поэтому ещё в первой серии мы заказали Raspberry Pi 4 в металлическом корпусе с 3.5" TFT-экранчиком. Pi приехал. Хозяин собрал, воткнул Ethernet, включил.

Красный LED горит. Зелёный (ACT) — нет.

> *«Этой малинкой никто не пользуется. И microSD-карта внутри пустая».*

Достали карту, посмотрели в Mac. Удивлены: **файловая система `bootfs` и партиция Linux** — на ней уже был записан Raspberry Pi OS Bookworm. «Пустая» была иронией: на карте всё было, кроме headless-настроек (SSH выключен, пользователь не создан). Pi загрузился бы и завис на экране первого запуска, ожидая клавиатуру с монитором, которых у нас не было.

Прошили заново через **Raspberry Pi Imager**:

- OS: Raspberry Pi OS (64-bit) with Desktop — на будущее, для TFT-экранчика
- Hostname: `bms-pi`
- SSH: enabled, password auth
- Wi-Fi: SSID + пароль (для интернета и SSH-доступа)
- Locale: Europe/London, layout GB

Вставили обратно, включили — через 51 секунду в mDNS появился `bms-pi.local → 192.168.1.153`. Загорелся зелёный ACT, начались бесконечные мигания. **Жив**.

Глава 6. Сеть, статика и mDNS на длинном таймауте

Сразу всплыла классика двухинтерфейсной жизни:

- **Wi-Fi (`wlan0`)** — домашний роутер, DHCP, 192.168.1.153, шлюз 192.168.1.254. Через это ходит SSH с Mac'a и интернет на самой Pi.
- **Ethernet (`eth0`)** — Titan-розетка в стене, статика `192.168.1.2/24`, без шлюза. Через это говорим с CIU на 192.168.1.151.

Проблема: обе сети живут в одной и той же подсети `192.168.1.0/24` (привет, строитель здания). Linux любит выбирать маршрут с меньшим metric, и без подсказки норовит отправить пакеты к CIU не через тот интерфейс.

Решение: явный маршрут `192.168.1.151/32 dev eth0` с приоритетом 100. Longest-prefix-match выигрывает любой /24, и весь трафик BACnet уходит туда, куда нужно. Всё остальное (домашняя сеть, GitHub, npm) — через Wi-Fi.

```
default via 192.168.1.254 dev wlan0 metric 600
192.168.1.0/24 dev wlan0 metric 600
192.168.1.0/24 dev eth0 metric 1000
192.168.1.151/32 dev eth0 metric 100 ← вот сюда смотри
```

Параллельно случилась мини-комедия. На запросе `ssh-copy-id pi@bms-pi.local` команда зависала намертво. Никаких подсказок, ничего. Оказалось — у mDNS-кэша на Mac случилась икота, и обращение по имени уходило в бесконечный DNS-таймаут. Лекарство: запустили по IP-адресу, ключ установился за секунду.

> *«Прошёл бесплатный двухчасовой курс по приоритету маршрутов в Linux. Не записывался, но получил сертификат».*

Глава 7. Дом → облако: SQLite, Vercel, Supabase

На Pi запустили обновлённый поллер. SQLite заменил CSV (с индексами, WAL, и колонкой `uploaded`). Появился systemd-юнит `bms-poller.service` — стартует при загрузке, перезапускается при сбоях. Параллельный `bms-uploader.service` раз в 30 секунд пушит свежие строки в облако.

📡 Titan / FCU × 3	🥧 Raspberry Pi	☁️ Vercel + Supabase
BACnet MS/TP, 65 объектов на цикл	poller (15 сек) → SQLite (WAL) uploader (30 сек) → HTTPS POST	Next.js API + Postgres cooling.olegkozyrev.com

Архитектура «Pi-first»: главное — чтобы данные сохранялись локально. Если интернет упадёт, uploader подождёт, накопит, потом догонит. Если упадёт Supabase — то же самое.

> **Грабли по дороге.** Поллер и uploader одновременно пишут/читают одну SQLite-базу. WAL-режим помогает с конкурентностью, но без `PRAGMA busy_timeout=10000` uploader моментально падал с «database is locked». Добавили — backlog в 519 строк улетел в облако за пару минут, и всё пришло в норму.

Глава 8. Next.js, Mantine и битва с CSS-слоями

Дашборд должен был быть простым: три карточки с температурами, статус системы, графики истории. На Next.js 16, с Mantine 9, с анимациями через `motion`, с фоном Canary Wharf и hero-секцией. Светлая тема, без переключателя.

На главной — **«Погода в доме»**. По центру — большие цифры RDU («В комнате»), под ними «Цель» и «Подача». Сверху всех — статус-баннер с пульсирующей точкой. Цвет

заголовка адаптируется к темноте/светлоте фотографии: на ночном Лондоне белый, на полуденном — чёрный.

Админка по паролю — за маленькой иконкой ключа в шапке. Внутри — графики (LineChart от @mantine/charts) и таблица всех записей с фильтром по комнате.

> **Откровение №5:** CSS-слои Tailwind v4 ловко перетирают стили Mantine, если импортировать обычный `styles.css`. Карточка ChartTooltip превращается в три SVG-текста, прилипших к точкам — драматично, но непригодно к жизни. Лечится одной строкой:
> `@import "@mantine/core/styles.layer.css";`
> `@import "@mantine/charts/styles.layer.css";`
> плюс `@layer tailwind, mantine;` в начале globals.css.

Дальше пошёл «маленький UI-марафон». Заголовок Hero съезжал. Фото фона не на всю страницу. На мобиле заголовок налезал на баннер. Тултипы графика рендерились дважды. SP подписали «Уставкой» — не по-человечески, переименовали в «Цель». Idle назвали «В уставке» — заменили на «Ожидает». Каждая итерация — коммит, push, Vercel автодеплой, скриншот, корректировка. К концу дня сайт стал выглядеть так, как и задумывался: спокойно, элегантно, на фоне ротации фото зданий Wood Wharf.

> *«Tailwind и Mantine жили в одной квартире и долго не могли решить, чьи стили будут победителями. Помог третий — @layer».*

Глава 9. Эпилог: дашборд в кармане

В 16:00 19 мая на телефоне открывается **cooling.olegkozyrev.com**. Три карточки. Спальня охлаждается (S2=13.5°C, клапан 60%). Гостиная и кабинет в норме. «Состояние системы — всё работает». На фоне плавно сменяются фото канарской набережной.

В админке за паролем — графики RDU и S2 по трём комнатам, период 1ч/6ч/24ч/7д. Линии гладкие, тултип показывает все три значения сразу и точное время. Y-ось зафиксирована 10–30°C — никакого «прыжка» масштаба от 19 до 19.5.

Pi в углу мигает. Поллер каждые 15 секунд опрашивает 65 BACnet-объектов на трёх FCU. Uploader отгружает в Supabase. Vercel рендерит дашборд на каждом запросе с last-fetch из Supabase. Поверх всего — Telegram-бот в планах, и нативное приложение на TFT 480×320 для самого Pi.

Что мы знаем точно после этой серии:

- Сбои не из-за полуночного расписания — из-за RDU=18.9°C
- ModuSat исправен, виноват общий чиллер здания
- Pi автономен, ребуты не страшны
- Данные в облаке, доступ откуда угодно

Что мы пока не знаем:

- Что именно делает чиллер ночью
- Поднимет ли SP=19°C в спальне срок жизни сбоев до нуля
- Что писать первому соседу, когда увидит дашборд

📡 → 🥧 → ☁️ → 📱

от BACnet'а к телефону — за два дня и одну бессонную ночь

Никакие данные не были изменены при проведении этого расследования.
Все клапаны и фанкойлы получили компенсацию.
Pi получил отдельную благодарность за зелёный LED.
© 19 мая 2026 года, лондонский вторник