Расширение системы температурного контроля спектрометрического магнита СПАСЧАРМ (М31)

Выполнил: М. В. Гани
Научный руководитель: П. А. Семёнов
(ведущий научный сотрудник НИЦ КИ-ИФВЭ)

НИЯУ МИФИ

25 декабря 2025 г.

УДК 539.12.01

Контекст и мотивация

- СПАСЧАРМ: спектрометрическая установка для исследований в адронных взаимодействиях.
- Импульсный анализ основан на измерении треков в магнитном поле.
- Магнит M31 — водоохлаждаемый; температурный мониторинг важен для безопасности и стабильной работы.
- Цель НИРС: расширить температурный контроль M31 за счёт второй линии датчиков (вторая «гирлянда») и корректно интегрировать новые каналы в EPICS.

Магнит M31 (иллюстрация из материалов по установке/системе управления).

Медленный контроль СПАСЧАРМ: EPICS-архитектура

- EPICS задаёт единый интерфейс к параметрам оборудования через PV (Process Variables).
- IOC на Raspberry Pi обеспечивает доступ к данным устройств (Modbus RTU / CAN) и публикует PV.
- Операторский уровень: CSS, аварийная сигнализация, архивирование и хранение параметров.

Задача расширения: добавить PV для оставшихся датчиков M31 (вторая линия 1-Wire на другом пине STM32), не ухудшив устойчивость IOC и время обновления.

Примечание: иллюстрации архитектуры и цепочки данных приведены на следующих слайдах.

Цепочка данных температуры M31 (фактическая схема)

- Raspberry Pi **не** работает с 1-Wire напрямую.
- Считывание датчиков: **STM32** $\rightarrow$ драйвер 1-Wire (**DS2480B**) $\rightarrow$ датчики температуры.
- Передача на верхний уровень: **Modbus RTU по RS-485** $\rightarrow$ IOC (asyn+modbus) $\rightarrow$ PV в EPICS.
- Вторая «гирлянда» — оставшиеся датчики, подключённые к **другой 1-Wire линии** (другой пин STM32).

Нижний уровень: особенности прошивки STM32

- 1-Wire реализован через DS2480B: поиск ROM (OWSearch), чтение scratchpad, преобразование в градусы.
- Modbus RTU: приём кадра с детектированием межкадрового интервала (типично 3.5 char) таймером.
- Важный инженерный момент: измерение температуры занимает время; при наивном опросе возможны конфликты с обслуживанием Modbus (нужны буферизация и/или планировщик).

Фрагмент пересчёта температуры (пример):

```
raw_temp = (data[1] << 8) | data[0];
real_temp = raw_temp * 0.0625f; // 12-
    bit шаг DS18B20
```

Ключевая идея расширения:

- добавить вторую 1-Wire линию;
- расширить карту регистров Modbus;
- обеспечить стабильный опрос обеих линий.

Верхний уровень: почему нужна асинхронность в IOC

Проблема синхронных обращений:

- блокировка record processing при ожидании ответа по RS-485;
- рост задержек обновления PV и «подвисания» интерфейса;
- лавинообразный рост тайм-аутов при деградации связи.

Как правильно:

- один поток/задача опроса (poller) $\rightarrow$ кэш измерений;
- PV читают *последнее валидное значение* без блокировок;
- статусы/алармы отражают качество связи и достоверность данных.

Что уже есть в прототипном IOC: связка `asyn` + `modbus` (настройка порта/тайм-аутов).

```
drvAsynSerialPortConfigure("SER","/
    dev/ttyUSB0",0,0,0)
asynSetOption("SER",0,"baud","9600"
    )
drvModbusAsynConfigure("TEMP","SER"
    ,1,3,0,84,0,1000,"Modbus")
```

Проект расширения на M31: PV и карта каналов

- Существующие датчики покрывают часть обмоток; вторая линия закрывает оставшиеся точки измерения.
- Предлагается разделить PV по BUS1/BUS2 (две 1-Wire линии STM32), сохранив единый стиль именования.

Канал	PV (пример)	Комментарий
1	M31MAGN:TEMP:BUS2:CH01	Обмотка/контур из «второй половины»
2	M31MAGN:TEMP:BUS2:CH02	...
⋮	⋮	⋮
24	M31MAGN:TEMP:BUS2:CH24	Последний датчик второй линии

(В IOC прототипа встречается префикс DINOMAGN для M29; в проекте для M31 используется отдельный префикс и отдельный блок регистров.)

Статус и ближайшие шаги (промежуточный отчёт)

Сделано на текущем этапе:

- анализ архитектуры EPICS/IOC и прототипных конфигов (asyn+modbus);
- анализ STM32-кода: 1-Wire через DS2480B, Modbus RTU, ограничения по времени опроса;
- подготовка схемотехнических материалов и проекта PV/карты каналов для M31 (BUS2).

План следующего этапа:

1. Прошивка STM32: вторая 1-Wire линия, расширение Modbus-регистров, диагностика ошибок.
2. IOC: переход на устойчивую схему «poller + кэш + статусы/алармы», добавление PV BUS2.
3. Стендовые испытания: нагрузка RS-485, задержки PV, сценарии отказов (обрыв/тайм-аут/ошибка CRC).
4. Монтаж оставшихся датчиков и интеграция в CSS/архиватор.

Спасибо за внимание!

Вопросы?