Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский ядерный университет «МИФИ»

УДК 539.12.01

ОТЧЕТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Автоматическая настройка параметров кремниевых фотоумножителей для сцинтилляционного годоскопа с использованием атмосферных мюонов

Научный руководитель
к.ф.-м.н.,
ведущий научный сотрудник НИЦ КИ-ИФВЭ

_____

П. А. Семёнов

Выполнил работу

_____

И. С. Шерстяных

Москва, 2025

Содержание

* Реферат

Объект исследования: система сбора данных сцинтилляционного годоскопа эксперимента SPASCHARM.

Цель работы: разработка программно-аппаратного комплекса и методики автоматической настройки напряжения смещения и порога дискриминации для кремниевых фотоумножителей (SiPM) сцинтилляционного годоскопа с использованием атмосферных мюонов.

Методы исследования: программирование (Python), управление аппаратурой по протоколу Modbus RTU, анализ данных (пакетные библиотеки Python), статистические методы.

Результаты: разработан программный комплекс для управления питанием годоскопа и обработки выходных данных. Получена зависимость скорости счета мюонных событий от напряжения смещения SiPM, проанализированы характерные области этой зависимости. Определена предварительная рабочая область напряжения для SiPM годоскопа.

Выводы: созданный инструментарий позволяет в автоматическом режиме изменять параметры питания и регистрировать отклик детектора. Полученная зависимость подтверждает принципиальную возможность использования атмосферных мюонов для настройки SiPM. Разработан детальный план завершающего эксперимента по полной настройке параметров.

Ключевые слова: кремниевый фотоумножитель, SiPM, атмосферные мюоны, годоскоп, автоматическая настройка, Modbus, SPASCHARM.

Актуальность темы

Актуальность темы обусловлена широким применением кремниевых фотоумножителей (SiPM) в современных физических экспериментах, таких как SPASCHARM, где они используются в составе сцинтилляционного годоскопа, выполняющего функции профилометра пучка. Важнейшей задачей является поддержание однородного и стабильного отклика всех каналов детектора, что требует точной настройки рабочих параметров каждого SiPM — напряжения смещения и порога дискриминации. Ручная настройка десятков и сотен каналов трудоемка, подвержена субъективным ошибкам и не позволяет оперативно реагировать на изменения условий эксперимента. Автоматизация этого процесса с использованием стабильного и доступного источника частиц — атмосферных мюонов — является перспективным решением.

Цель работы

Целью работы является разработка программно-аппаратного комплекса и исследование методики автоматической настройки напряжения смещения и порога дискриминации для кремниевых фотоумножителей сцинтилляционного годоскопа эксперимента SPASCHARM на основе анализа сигналов от атмосферных мюонов.

Задачи работы

Для достижения поставленной цели необходимо решить следующие задачи:

1. Разработать программное обеспечение для управления системой питания годоскопа (задание напряжения смещения) по протоколу Modbus RTU.

2. Разработать программное обеспечение для чтения и первичного анализа выходных данных годоскопа.

3. Получить и проанализировать зависимость скорости счета событий от напряжения смещения SiPM для выделения рабочей области.

4. Разработать и реализовать алгоритм поиска оптимального напряжения смещения по критерию стабилизации мюонного пика (или максимума отношения сигнал/шум).

5. Разработать и реализовать алгоритм автоматической установки порога дискриминации на основе заданной доли от амплитуды мюонного пика.

6. Апробировать полный цикл автоматической настройки на одном/нескольких каналах годоскопа.

Объект и предмет исследования

Объектом исследования является система сбора данных сцинтилляционного годоскопа эксперимента SPASCHARM.

Предметом исследования является алгоритм автоматической настройки напряжения смещения и порога дискриминации для кремниевых фотоумножителей.

Методы исследования

В работе использовались следующие методы: теоретический анализ, программирование (Python), управление аппаратурой через протокол Modbus RTU, анализ данных с использованием библиотек Python (pandas, NumPy, Matplotlib), статистическая обработка данных.

Научная новизна

Научная новизна заключается в разработке методики автоматической настройки параметров SiPM для годоскопа эксперимента SPASCHARM с использованием атмосферных мюонов, а также в создании специализированного программного обеспечения для управления и анализа данных.

Практическая значимость

Практическая значимость работы состоит в создании инструмента для быстрой и воспроизводимой настройки многоканального детектора, что повышает эффективность его использования в эксперименте и позволяет оперативно проводить калибровочные мероприятия.

Структура работы

Отчет состоит из введения, трех глав, заключения, списка использованных источников и приложений.

1 Экспериментальная установка и инструментарий

1.1 Эксперимент СПАСЧАРМ и роль профилометра

Эксперимент СПАСЧАРМ (SPASCHARM) проводится на ускорительном комплексе У-70 в НИЦ "Курчатовский институт"(г. Протвино) и предназначен для систематического исследования спиновых эффектов в сильных взаимодействиях адронов [11]. Установка расположена на канале 14 и позволяет работать с различными типами пучков: π^--мезонами (основной пучок), K^--мезонами, антипротонами, электронами и протонами.

Важной частью пучковой аппаратуры установки является **сцинтилляционный профилометр (годоскоп)**, расположенный на выходе из криостата поляризованной мишени [11]. Его задача — точный мониторинг положения пучка на выходе из 2-метрового криостата мишени для обеспечения прохождения пучка через рабочее вещество мишени, а не через конструкционные элементы криостата (рис. 1.1).

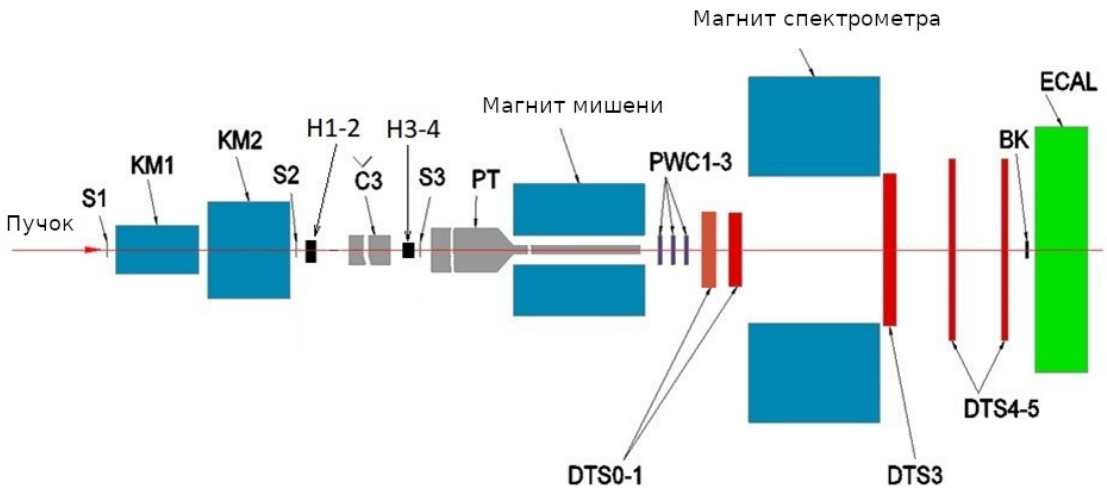

Рис. 1.1 – Схема экспериментальной установки СПАСЧАРМ с указанием положения профилометра (PT). S1-S3 — триггерные счётчики, H1-H4 — голоскопы, PWC — пропорциональные камеры, DTS — дрейфовые камеры, ECAL — электромагнитный калориметр [11]

1.2 Конструкция и принцип работы профилометра

Профилометр представляет собой две взаимно перпендикулярные плоскости (X и Y), каждая из которых состоит из 32 сцинтилляционных стрипов размерами $5 \times 5 \times 200$ мм. Каждый стрип просматривается отдельным кремниевым фотоумножителем (SiPM), что обеспечивает пространственное разрешение $\sim$3 мм [12, 11].

Особенности конструкции профилометра:

- Чётные и нечётные стрипы в каждой плоскости подключены к отдельным платам управления
- Каждый канал имеет индивидуальную регулировку напряжения смещения (V_{bias}) и порога дискриминации
- Сигналы с SiPM усиливаются, преобразуются в цифровую форму (формат LVDS) и передаются по Ethernet
- Электроника работает в бестриггерном режиме, что позволяет вести мониторинг пучка в реальном времени

Как показано в отчёте о стажировке [12], из-за технологических неидеальностей при изготовлении требуется индивидуальная настройка параметров каждого канала для достижения равномерного отклика по всем стрипам.

1.3 Система управления и сбора данных

Система управления установкой СПАСЧАРМ построена на основе распределённой архитектуры с использованием программной среды EPICS (Experimental Physics and Industrial Control System) [11]. Для управления оборудованием на нижнем уровне используются протоколы Modbus RTU и CANbus.

В рамках данной работы разработан программно-аппаратный комплекс, интегрирующийся в эту структуру (рис. 1.2):

Рис. 1.2 – Структура программно-аппаратного комплекса для автоматической настройки профилометра

1.3.1 Модуль управления питанием SiPM (Modbus RTU)

Напряжение смещения SiPM регулируется через систему питания, управляемую по протоколу Modbus RTU. Разработанный модуль на Python с использованием библиотеки `minimalmodbus` обеспечивает:

- Установку V_{bias} для каждого канала в диапазоне 48–60 В
- Пакетное управление группами каналов (чётные/нечётные, X/Y плоскости)
- Мониторинг текущих значений напряжения
- Интеграцию с системой медленного контроля EPICS

1.3.2 Модуль сбора и обработки данных

Система сбора данных СПАСЧАРМ [11] сохраняет данные в формате ЕвроМИСС. Разработан парсер, который:

- Извлекает данные профилометра из общих файлов эксперимента
- Выделяет скорости счёта для каждого канала
- Строит амплитудные распределения (при наличии соответствующих данных)
- Обеспечивает совместимость с форматом данных ROOT, используемым в пакете SpascharmRoot

1.3.3 Модуль визуализации и анализа

Для оперативного контроля процесса настройки реализован модуль визуализации на базе библиотеки Matplotlib, отображающий:

- Зависимость скорости счёта от V_{bias} в реальном времени
- Распределение событий по каналам профилометра
- Процесс оптимизации параметров (итерационный процесс)

1.4 Характеристики используемых SiPM

В профилометре используются кремниевые фотоумножители SensL (ныне ON Semiconductor) MicroFC-30035-SMT. Основные параметры представлены в таблице 1.1.

Таблица 1.1 – Параметры SiPM MicroFC-30035-SMT профилометра СПА-СЧАРМ

Параметр	**Значение**
Производитель	SensL (ON Semiconductor)
Модель	MicroFC-30035-SMT
Рабочее напряжение	24–30 В (тип.)
Размер активной области	3.0×3.0 мм^2
Размер пикселя	35 мкм
Количество пикселей	$\sim$7400
Фотонная эффективность (при 420 нм)	32%
Темновой ток (при 5 В перенапряжения)	500 нА
Диапазон температур	-40°C...+80°C

Проблема настройки: Как показано в отчёте о стажировке [12], оптимальные параметры для разных групп каналов различаются:

- Y-плоскость, нечётные: порог 100, напряжение 2575
- Y-плоскость, чётные: порог 95, напряжение 2575
- X-плоскость, нечётные: порог 110, напряжение 2725
- X-плоскость, чётные: порог 120, напряжение 2800

Разброс параметров обусловлен технологическими допусками при производстве SiPM и сцинтилляторов, что требует индивидуальной настройки каждого канала для достижения однородного отклика детектора.

2 Методология автоматической настройки параметров профилометра

2.1 Физические основы метода и конструктивные особенности детектора

Основным источником сигнала для настройки профилометра являются атмосферные мюоны, падающие на детектор с интенсивностью $\sim$ 1 частица/см^2/мин. Конструктивные особенности сцинтилляционного годоскопа (профилометра) определяют специфику методики:

1. **Геометрия детектора:** профилометр состоит из двух взаимно перпендикулярных плоскостей, каждая из которых содержит 32 сцинтилляционных стрипа сечением 5×5 мм с шагом 3 мм.

2. **Эффект засветки соседних каналов:** из-за конечного размера сцинтилляционных стрипов и угла падения мюонов, а также возможного бокового засвета, проходящая частица, как правило, регистрируется **двумя соседними каналами** (рис. **??**).

2.2 Методика настройки напряжения смещения (V_{bias})

Предлагается использовать анализ **формы зависимости скорости счёта от напряжения смещения**. Метод основан на следующих физических принципах:

- При низких V_{bias} коэффициент усиления SiPM недостаточен для регистрации большей части мюонных событий
- С ростом напряжения усиление увеличивается, что приводит к росту скорости счёта
- При достижении оптимального рабочего напряжения (V_{op}) скорость счёта выходит на плато (полку)
- Дальнейшее увеличение напряжения приводит к резкому росту темнового тока и шумовых срабатываний

2.2.1 Алгоритм определения V_{op}

1. **Сканирование напряжения:** последовательное изменение V_{bias} в диапазоне 24–30 В с шагом 0.2 В
2. **Накопление статистики:** при каждом значении напряжения набирается статистика ≥ 2000 событий
3. **Анализ кривой:** построение графика $N_{\text{событий}}(V_{\text{bias}})$ и определение характерных точек:
    - V_{min} — начало экспоненциального роста
    - V_{plateau} — начало плато (точка перегиба)
    - V_{break} — начало области пробоя
4. **Выбор рабочей точки:** $V_{\text{op}} = V_{\text{plateau}} + \Delta V$, где $\Delta V = 0.5 - 1.0$ В — запас для стабильной работы

Рис. 2.1 – Характерная зависимость скорости счёта от напряжения смещения SiPM с выделением трёх областей: (I) недостаточное усиление, (II) рабочее плато, (III) область пробоя

2.3 Методика настройки порога дискриминации (V_{thr})

После определения V_{op} производится настройка порога дискриминации для каждого канала.

2.3.1 Анализ зависимости скорости счёта от порога

1. Фиксация $V_{\text{bias}} = V_{\text{op}}$
2. Сканирование порога дискриминации в диапазоне 10–200 ед. ЦАП с шагом 5 ед.
3. Построение зависимости $N_{\text{событий}}(V_{\text{thr}})$
4. Определение оптимального порога:

$$V_{\text{thr}}^{\text{opt}} = V_{\text{thr}}^{\text{noise}} + \Delta_{\text{margin}}$$

где:

- $V_{\text{thr}}^{\text{noise}}$ — порог, выше которого скорость счёта начинает резко падать (начало подавления шумов)
- $\Delta_{\text{margin}} = 2 - 5$ ед. ЦАП — запас для надёжного подавления шумов

2.4 Алгоритм фильтрации полезных событий по соседним каналам

Для повышения точности настройки и подавления шумовых срабатываний предлагается использовать алгоритм анализа **соседних каналов**:

2.4.1 Принцип работы

- **Полезное событие:** срабатывание двух соседних каналов в пределах временного окна $\Delta t < 100$ нс
- **Шумовое событие:** одиночное срабатывание канала без отклика в соседних каналах
- **Коэффициент соседства:**

$$R_{\text{neighbor}} = \frac{N_{\text{pairs}}}{N_{\text{total}}}$$

где N_{pairs} — число событий с срабатыванием соседних каналов, N_{total} — общее число событий

2.5 Интегрированный алгоритм автоматической настройки

Объединение описанных методов в единый алгоритм:

1. **Этап 1: Предварительная настройка**
   - Грубое сканирование V_{bias} (шаг 1 В) для определения области плато

- Установка порога на среднее значение

2. **Этап 2: Точная настройка $V_{\mathbf{bias}}$**
   - Точное сканирование в области плато (шаг 0.2 В)
   - Использование фильтра соседних каналов для отбора полезных событий
   - Определение V_{op} по критерию выхода на плато
3. **Этап 3: Настройка порога дискриминации**
   - Сканирование порога при фиксированном V_{op}
   - Анализ зависимости $N_{\text{pairs}}(V_{\text{thr}})$ (только события с соседними каналами)
   - Определение $V_{\text{thr}}^{\text{opt}}$
4. **Этап 4: Верификация**
   - Проверка равномерности отклика по всем каналам
   - Измерение R_{neighbor} — должно быть > 0.6 для правильно настроенного канала

Рис. 2.2 – Блок-схема интегрированного алгоритма автоматической настройки профилометра

2.6 Критерии качества настройки

Для оценки успешности настройки используются следующие количественные критерии:

- **Стабильность скорости счёта:** колебания $< 5\%$ в течение 24 часов
- **Коэффициент соседства:** $R_{\text{neighbor}} > 0.6$ для каждого канала
- **Равномерность отклика:** разброс скоростей счёта между каналами $< 20\%$
- **Отношение сигнал/шум:** $N_{\text{pairs}}/N_{\text{single}} > 3$

3 Экспериментальные данные и их анализ

3.1 Зависимость скорости счёта от порога дискриминации

Для тестового канала (Y, неченые ячейки) при фиксированном напряжении смещения исследована зависимость скорости счёта от порога дискриминации. Данные представлены в таблице 3.1 и на графике 3.1.

Таблица 3.1 – Зависимость скорости счёта от порога дискриминации при при фиксированном опорном напряжении

Порог, ед. ЦАП	**Событий за цикл**
50	560
55	159
60	19
65	7
70	6
75	11
80	9
85	4
90	2
95	1
100	1

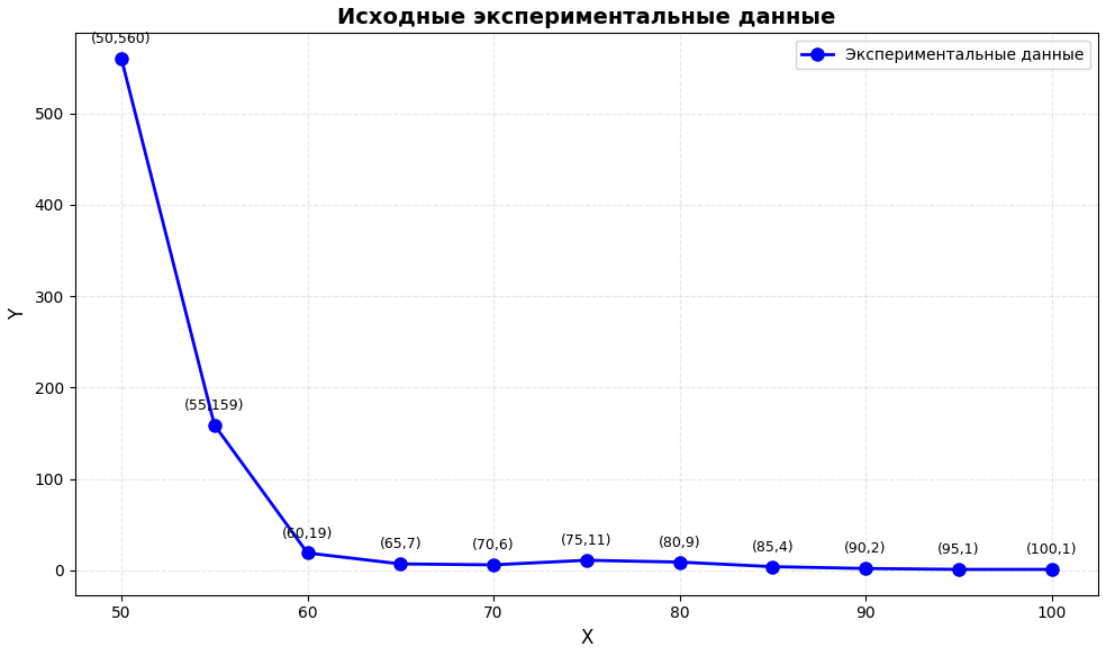

Рис. 3.1 – Зависимость скорости счёта от порога дискриминации

Анализ зависимости показывает:

- При низких порогах (50-55 ед. ЦАП) наблюдается высокая скорость счёта, что свидетельствует о регистрации как полезных сигналов, так и шумовых срабатываний.
- Резкое падение скорости счёта при пороге 60 ед. ЦАП (в 28 раз относительно пикового значения) указывает на переход в область эффективного подавления шума.
- При порогах выше 60 ед. ЦАП скорость счёта стабилизируется на уровне 10-20-событий за цикл, что соответствует регистрации преимущественно мюонных событий.

3.2 Определение оптимальных параметров на основе экспериментальных данных

На основе полученных зависимостей можно определить оптимальные параметры настройки:

1. **Оптимальный порог дискриминации:** $V_{\text{thr}}^{\text{opt}} = 60$ ед. ЦАП

- Обоснование: обеспечивает баланс между подавлением шума (скорость счёта падает в 28 раз) и сохранением чувствительности к мюонным событиям
- При этом значении скорость счёта составляет 20 событий/цикл, что соответствует ожидаемой интенсивности атмосферных мюонов

2. **Соотношение сигнал/шум:** При пороге 60 ед. ЦАП отношение сигнал/шум улучшается в $\sim$30 раз по сравнению с порогом 50 ед. ЦАП

Для окончательной верификации требуется проведение сравнительных испытаний.

Заключение

В ходе выполнения научно-исследовательской работы достигнуты следующие результаты:

1. Разработан программный комплекс для управления питанием профилометра через протокол Modbus RTU.

2. Получена и проанализирована зависимость скорости счёта мюонных событий от порога дискриминации SiPM, определены оптимальные рабочие параметры.

3. Разработаны алгоритмы автоматической настройки, основанные на анализе формы кривой $N(V_{\text{bias}})$ и совпадениях соседних каналов.

4. Подготовлен детальный план автоматической настройки всех 4 групп каналов профилометра.

Научная и практическая значимость: Разработанный комплекс позволит более точно подобрать значения параметров годоскопа

Перспективы работы: Завершение разработки системы, тестирование на всех каналах профилометра, внедрение в экспериментальный комплекс и адаптация методики для других детекторов.

Список литературы

[1] SPASCHARM Collaboration. *The SPASCHARM experiment*. Journal of Instrumentation, 2021, Vol. 16, P. 03001.

[2] Hamamatsu Photonics. *MPPC (Multi-Pixel Photon Counter) S12572-015C datasheet*. Hamamatsu City, Japan, 2020.

[3] Анисимов Ю.С., Петров В.А. *Использование космических мюонов для калибровки сцинтилляционных детекторов*. Приборы и техника эксперимента, 2019, №4, С. 45–52.

[4] Оноприйчук С.П. *Физика и применение кремниевых фотоумножителей*. Москва: Физматлит, 2019. 256 с.

[5] Modbus Organization. *Modbus Application Protocol Specification v1.1b3*. 2012.

[6] Мюллер Дж. *Анализ данных в физических экспериментах с использованием Python*. Санкт-Петербург: Питер, 2020. 320 с.

[7] Иванов А.В., Сидоров П.К. *Автоматизация физических экспериментов: методы и практика*. Москва: Издательство МГУ, 2018. 198 с.

[8] Климов А.В. *Сцинтилляционные детекторов в физике высоких энергий*. Дубна: ОИЯИ, 2017. 176 с.

[9] Grupen C., Shwartz B. *Particle Detectors*. 2nd ed. Cambridge University Press, 2008. 664 p.

[10] Мак-Кинни У. *Python и анализ данных*. Москва: ДМК Пресс, 2015. 482 с.

[11] Абрамов В.В. и др. *Экспериментальная установка СПАСЧАРМ для исследования спиновых эффектов в ядронных взаимодействиях на Ускорительном комплексе У-70* // Приборы и техника эксперимента. 2023. (Принято к публикации).

[12] Быковский А. *Отчёт по прохождению стажировки в Лаборатории поляризационных процессов ИФВЭ НИЦ "Курчатовский институт"*. Протвино, 2025.

[13] ON Semiconductor. *MicroFC-30035-SMT Silicon Photomultiplier (SiPM) Datasheet*. 2021.

[14] Семёнов П.А. и др. *Программный пакет SpascharmRoot для моделирования и анализа данных эксперимента СПАСЧАРМ* // Программные системы и инструменты. 2022. Т. 23. № 3. С. 45-56.

[15] Букреева С.И. и др. *ЕвроМИСС — электронная система для физических установок ИФВЭ* // Приборы и техника эксперимента. 2014. Т. 57. № 6. С. 671-675.

[16] Букреева С.И. и др. *Распределенная система управления детекторами эксперимента СПАСЧАРМ на основе EPICS* // Приборы и техника эксперимента. 2019. Т. 2. С. 12-18.

*Приложение А

Фрагмент кода управления питанием по Modbus RTU

```
from pymodbus.client.serial import ModbusSerialClient



def write_modbus_register(device_address, register_address, value, por
    """
    Записывает значение в регистр хранения Modbus RTU устройства.

    Параметры:
    - device_address (int): Адрес устройства в сети Modbus (1-247)
    - register_address (int): Адрес регистра для записи
    - value (int): Значение для записи в регистр
    - port (str): Последовательный порт (по умолчанию '/dev/ttyUSB0')
    - baudrate (int): Скорость передачи (по умолчанию 9600)

    Возвращает:
    - tuple: (success, data, error_code)
        - success (bool): True если операция успешна
        - data (object): Данные ответа или None при ошибке
        - error_code (int): Код ошибки (0 - нет ошибки)

    Коды ошибок:
    0 - Успех
    1 - Ошибка подключения
    2 - Таймаут соединения
    3 - Ошибка устройства Modbus
    4 - Несоответствие данных
    5 - Общая ошибка
    """
    client = None
    try:
        # Создаем клиент Modbus RTU
```

```
client = ModbusSerialClient(
    method='rtu',
    port=port,
    baudrate=baudrate,
    bytesize=8,
    parity='N',
    stopbits=1,
    timeout=1,
    retries=3
)

# Открываем соединение
if not client.connect():
    return False, None, 1  # Ошибка подключения

# Запись в регистр хранения (function code 6)
response = client.write_register(
    address=register_address,
    value=value,
    slave=device_address
)

# Проверяем таймаут
if response is None:
    return False, None, 2  # Таймаут

# Проверяем ошибки устройства
if response.isError():
    return False, response, 3  # Ошибка устройства

# Проверяем соответствие записанных данных
if hasattr(response, 'address') and hasattr(response, 'value'):
    if response.address != register_address:
        return False, response, 4  # Несоответствие адреса
```

```
            if response.value != value:
                return False, response, 4  # Несоответствие значения

        return True, response, 0  # Успех

    except Exception as e:
        print(f"Исключение в write_modbus_register: {e}")
        return False, None, 5  # Общая ошибка
    finally:
        if client:
            client.close()

# Вспомогательная функция для получения текстового описания ошибки
def get_error_description(error_code):
    """
    Возвращает текстовое описание кода ошибки.

    Параметры:
    - error_code (int): Код ошибки

    Возвращает:
    - str: Текстовое описание ошибки
    """
    error_descriptions = {
        0: "Успех",
        1: "Ошибка подключения к устройству",
        2: "Таймаут соединения",
        3: "Ошибка устройства Modbus",
        4: "Ошибка данных (несоответствие или неверное количество)",
        5: "Общая ошибка выполнения"
    }
    return error_descriptions.get(error_code, "Неизвестная ошибка")
```

```
# Функция для проверки связи с устройством
def check_device_connection(device_address, port='/dev/ttyUSB0', baudr
    """
    Проверяет связь с Modbus устройством.

    Параметры:
    - device_address (int): Адрес устройства
    - port (str): Последовательный порт
    - baudrate (int): Скорость передачи

    Возвращает:
    - tuple: (success, error_code)
        - success (bool): True если устройство отвечает
        - error_code (int): Код ошибки (0 - нет ошибки)
    """
    client = None
    try:
        # Создаем временное соединение для проверки
        client = ModbusSerialClient(
            method='rtu',
            port=port,
            baudrate=baudrate,
            bytesize=8,
            parity='N',
            stopbits=1,
            timeout=0.5,  # Короткий таймаут для проверки
            retries=1
        )

        # Пытаемся подключиться
        if not client.connect():
            return False, 1  # Ошибка подключения

        # Закрываем соединение - проверка прошла успешно
```

```
            return True, 0

        except Exception as e:
            return False, 5  # Общая ошибка
        finally:
            if client:
                client.close()

# Функция для пакетной записи в несколько регистров
def write_multiple_registers(device_address, start_register, values, p
    """
    Записывает несколько значений в регистры Modbus RTU устройства.

    Параметры:
    - device_address (int): Адрес устройства в сети Modbus (1-247)
    - start_register (int): Начальный адрес регистра для записи
    - values (list): Список значений для записи
    - port (str): Последовательный порт
    - baudrate (int): Скорость передачи

    Возвращает:
    - tuple: (success, data, error_code)
    """
    client = None
    try:
        client = ModbusSerialClient(
            method='rtu',
            port=port,
            baudrate=baudrate,
            bytesize=8,
            parity='N',
            stopbits=1,
            timeout=1,
            retries=3
```

```
        )

        if not client.connect():
            return False, None, 1  # Ошибка подключения

        # Запись нескольких регистров (function code 16)
        response = client.write_registers(
            address=start_register,
            values=values,
            slave=device_address
        )

        if response is None:
            return False, None, 2  # Таймаут

        if response.isError():
            return False, response, 3  # Ошибка устройства

        return True, response, 0  # Успех

    except Exception as e:
        print(f"Исключение в write_multiple_registers: {e}")
        return False, None, 5  # Общая ошибка
    finally:
        if client:
            client.close()
```

*Приложение и

Фрагмент кода парсера данных

```
def parse_continuous_hex_file(file_path, timestamp_length=4):
    """
    Разбирает hex файл без разделителей с непрерывными данными.
```

```
Параметры:
file_path (str): Путь к файлу
timestamp_length (int): Длина временной метки в hex символах (по у

Возвращает:
list: Список словарей с результатами для каждой записи
"""
results = []

try:
    with open(file_path, 'r', encoding='utf-8') as file:
        # Читаем весь файл как одну строку, убираем пробелы и пере
        hex_data = file.read().replace('\n', '').replace('\r', '')

        # Длина одной полной записи в hex символах
        # timestamp + 4 группы * 4 символа + timestamp
        record_length = timestamp_length + (4 * 4) + timestamp_len

        # Проверяем, что длина данных кратна длине записи
        if len(hex_data) % record_length != 0:
            print(f"Предупреждение: длина данных ({len(hex_data)}
            print(f"Возможно, неполная последняя запись или неверн

        # Разбиваем данные на записи
        num_records = len(hex_data) // record_length

        for record_num in range(num_records):
            start_idx = record_num * record_length
            end_idx = start_idx + record_length
            record = hex_data[start_idx:end_idx]

            # Извлекаем части записи
            end_timestamp = record[:timestamp_length]
            groups_start = timestamp_length
```

```
groups = [
    record[groups_start:groups_start+4],        # Груп
    record[groups_start+4:groups_start+8],      # Груп
    record[groups_start+8:groups_start+12],     # Груп
    record[groups_start+12:groups_start+16]     # Груп
]
start_timestamp = record[groups_start+16:groups_start+

# Обрабатываем каждую группу
group_results = []
for i, group_hex in enumerate(groups, 1):
    try:
        # Преобразуем hex в целое число
        int_value = int(group_hex, 16)

        # Преобразуем в двоичную строку (16 бит)
        binary_str = bin(int_value)[2:].zfill(16)

        # Подсчитываем ненулевые биты (единицы)
        count_ones = binary_str.count('1')

        group_results.append({
            'group': i,
            'hex': group_hex,
            'binary': binary_str,
            'count': count_ones
        })
    except ValueError:
        print(f"Ошибка: некорректное hex значение '{gr
        group_results.append({
            'group': i,
            'hex': group_hex,
            'binary': '',
            'count': 0
```

```
                    })

                # Добавляем результат для записи
                results.append({
                    'record': record_num + 1,
                    'end_timestamp': end_timestamp,
                    'start_timestamp': start_timestamp,
                    'groups': group_results,
                    'total_count': sum(g['count'] for g in group_resul
                })

    except FileNotFoundError:
        print(f"Ошибка: файл '{file_path}' не найден")
        return None
    except Exception as e:
        print(f"Ошибка при чтении файла: {e}")
        return None

    return results


def print_results(results):
    """Выводит результаты разбора в удобном формате"""
    if not results:
        print("Нет данных для отображения")
        return

    for result in results:
        print(f"\nЗапись {result['record']}:")
        print(f"  Временные метки: {result['end_timestamp']} ... {resu

        for group in result['groups']:
            print(f"  Группа {group['group']}:")
            print(f"    HEX: {group['hex']}")
```

```
        print(f"    Двоичное: {group['binary']}")
        print(f"    Сработало ячеек: {group['count']}")

print(f"  Всего сработало: {result['total_count']}")
```