Федеральное государственное бюджетное учреждение науки Институт ядерных исследований Российской академии наук «ИЯИ РАН»

На правах рукописи

Чернов Василий Геннадьевич

Разработка распределенной системы сбора данных и анализ формы импульса событий на установке «Троицк ню-масс»

01.04.01 — «Приборы и методы экспериментальной физики»

Диссертация
на соискание ученой степени
кандидата физико-математических наук

Научный руководитель:
кандидат физико-математических наук
Нозик Александр Аркадьевич

Москва — 2019

Оглавление

Стр.

Стр.

Введение

Актуальность темы

На сегодняшний день нейтринные исследования относятся к одной из самых активных и перспективных областей физики элементарных частиц. Экспериментальное открытие нейтринных осцилляций и смешиваний [1][2] свидетельствует о выходе физики нейтрино за рамки Стандартной модели. Изучение свойств этой группы фундаментальных частиц открывает возможность прямого исследования явлений новой физики.

Экспериментально полученные ограничения на параметры нейтрино позволяют предположить существование еще одного вида частиц - стерильных (обладающих правой хиральностью) нейтрино. Большинство расширений Стандартной модели предполагают существование одного или нескольких видов такой частицы с модельно-зависимой массой. Аргументы в пользу гипотезы существования стерильных нейтрино:

- существование стерильного нейтрино естественным образом объясняет наличие массы активных нейтрино;
- для всех остальных фермионов были обнаружены как левосторонние, так и правосторонние частицы.

Эксперимент «Троицк ню-масс» с 2012 года присоединился к поискам стерильного нейтрино [3]. Установка, несмотря на изначальное проектирование под измерение массы активного нейтрино, может быть использована и для поиска стерильных нейтрино с массой в диапазоне единиц кэВ. Эта область масс представляет интерес, т. к. стерильное нейтрино с массой порядка нескольких кэВ является одним из кандидатов на роль частиц темной материи [4]. Поиск стерильных нейтрино требует модификации установки. Необходимые улучшения описаны в [5]. Данная работа касается одного из основных необходимых улучшений - повышения чувствительности системы считывания сигнала.

Целью данной работы является совершенствование системы сбора данных установки «Троицк ню-масс» и разработка алгоритмов разделения наложений при непрерывной оцифровке.

Для достижения поставленной цели необходимо было решить следующие **задачи**:

1. Провести общую модернизацию системы сбора данных. Разработать модульную архитектуру, позволяющую проводить изолированную разработку и отладку отдельных подсистем, в частности считывания сигнала. Адаптировать существующий алгоритм управления установкой под новую архитектуру. Добавить инструменты контроля качества набора в реальном времени.

   Необходимость разработки распределенной архитектуры вызвана масштабом системы сбора данных. Работа по уменьшению мертвого времени требует тестирования и отладки методов считывания сигнала. В силу сложности системы единственный способ эффективной разработки программного обеспечения - разбиение его на независимые подсистемы и разработка каждого модуля по отдельности. Дополнительными преимуществами модульности являются: возможность быстрой замены и одновременного использования нескольких вариантов считывающей электроники, возможность стандартизации интерфейса взаимодействия между подсистемами и использование единого транспортного протокола.
2. Провести исследование доступных вариантов альтернативного считывания сигнала. Оценить возможность использования в качестве замены исходной подсистемы считывания для установки «Троицк ню-масс». Встроить наиболее подходящий вариант в систему сбора данных. Реализовать для системы новый модуль, осуществляющий автоматизированный контроль считывания. Протестировать созданную подсистему и убедиться в ее корректной работе.
3. Разработать алгоритмы обработки считанных данных, позволяющие работать с сигналами с высокой скоростью счета.

Научная новизна:

1. С учетом современных подходов к проектированию сложных программных комплексов[6][7] разработана распределенная система сбора данных установки «Троицк ню-масс». В основе архитектуры лежит распределение обработки по независимым модулям, каждый из которых работает независимо и контролирует отдельную подсистему обработки. Модули имеют единообразный программный интерфейс и взаимодействуют через стандартный стек TCP/IP. Аппаратно модернизация основана на использовании встраиваемых ПК-контроллеров

CCPC7[8] производства ОИЯИ (г. Дубна) вместо обычных контроллеров КАМАК, подключающихся к ПК.

2. В рамках проектирования системы сбора данных разработан оригинальный формат передачи и хранения данных, оптимизированный под задачи экспериментальной физики[9].
3. Разработан оригинальный алгоритм разделения наложенных сигналов, основанный на форме событий. С помощью алгоритма получено эффективное мертвое время порядка 0.9 мкс при длине сигнала 6 мкс и одного канала оцифровки 320 нс. Помимо качества разделения, существенной особенностью алгоритма является отсутствие привязки к конкретной форме сигнала.

Практическая значимость. Распределенные системы контроля и сбора данных представляют существенный интерес для физических экспериментов. Успешный опыт использования этой концепции в эксперименте «Троицк ню-масс» будет применен при разработке новых систем. Распределенная система хранения данных и обмена сообщениями также представляет интерес как для экспериментальной физики, так и для промышленного использования.

Использование разработанных алгоритмов по разделению наложенных сигналов в рамках эксперимента «Троицк ню-масс» позволило добиться уменьшения эффективного мертвого времени в семь раз и, таким образом, обеспечить возможность работы на скорости счета вплоть до 50-60 кГц на канал, что позволит проводить измерения бета-спектра в широком диапазоне и с большой статистикой. Существенным является то, что алгоритмы не привязаны к конкретной форме сигнала и могут быть адаптированы для использования в других экспериментах.

Методология и методы исследования. При проектировании системы сбора данных проводилось исследование архитектуры комплексных систем с открытым исходным кодом. Проводился анализ подходов в существующих проектах, определялись их преимущества и возможность применения в разрабатываемой системе сбора. Для реализации компонент системы проводился поиск готового и поддерживаемого инструментария с открытым исходным кодом. В целом критериями выбора того или иного подхода и фреймворка были: его популярность, простота использования и степень распространенности в проектах.

При разработке методов разделения наложенных сигналов был проведен анализ существующих решений, изложенных в статьях и препринтах. Для

оценки качества работы алгоритмов был разработан генератор событий, симулирующий реальный сигнал детектора. Адекватность работы генератора оценивалась с помощью сравнения статистических параметров с реальными данными. Техника определения метрик качества генерации данных и алгоритмы обработки описаны в соответствующих главах работы.

Основные положения, выносимые на защиту:

1. Проведена общая модернизация системы сбора данных установки «Троицк ню-масс». Выполнен переход от монолитной архитектуры исходной системы к микросервисной архитектуре. Основной код обработки был переписан на C++/Qt. Модернизированная система разбита на три независимых модуля, отвечающих за: считывание сигнала с детектора, управление напряжением спектрометра и проведение набора по сценарию. Модули имеют единообразный программный интерфейс и взаимодействуют через стек TCP/IP. В основе аппаратной модернизации лежит использование ПК-контроллеров крейтов КАМАК CCPC7 для управления подсистемами установки. Каждый модуль представляет собой программный сервер, который устанавливается в CCPC7 и осуществляет контроль аппаратуры через магистраль крейта, RS-232, RS-485 интерфейсы и другие специфические соединения.
2. Осуществлен переход на запись и обработку непрерывных кадров сигнала. Для модернизированной системы сбора разработан модуль считывания с помощью АЦП Лан10-12PCI. В процессе записи плата последовательно сбрасывает кадры максимальной длины по программному триггеру. Каждый последующий кадр сбрасывается сразу после сохранения предыдущего. Таким образом, сохраняется непрерывная оцифровка сигнала с пропусками на время сбросов. Данный подход позволяет решить проблему аппаратного мертвого времени, возникающую при стандартном наборе по триггеру, и проводить более сложную обработку сигнала в офлайн-режиме.
3. В рамках сотрудничества с группой TRISTAN[10] из Института физики имени Макса Планка в Мюнхене, к системе сбора данных «Троицк ню-масс» был подключен прототип детектора TRISTAN, разрабатываемого для KATRIN[11]. Управление детектором осуществлялось через устройство обработки сигнала DANTE производства XGLab, Италия. Для него разработан эмулятор сигналов, работающий по предостав-

ленной спецификации. При помощи эмулятора создан программный пакет, обеспечивающий взаимодействие с DANTE, управление считыванием, а также хранение событий вместе со специфическими для этой системы метаданными. При помощи этой системы проведены два сеанса измерений на установке с использованием детектора TRISTAN, по результатам которых была подготовлена совместные статьи [12] [13].

4. Разработаны алгоритмы выделения параметров событий (амплитуды и положения по времени) из непрерывных кадров с фиксированной длиной, полученных с платы Лан10-12PCI. Для данных «Троицк ню-масс» при выделении параметров событий удалось добиться эффективного мертвого времени порядка 0.9 мкс для средней длины сигнала 6 мкс при шаге оцифровки 320 нс. Мертвое время при использовании исходной аппаратной обработки событий составляло около 7 мкс. Учет формы импульса при обработке позволил исключить систематическую ошибку восстановления амплитуд, вызванную наложением на хвосты предыдущих событий. Проведено обобщение алгоритмов для использования с произвольной формой импульса. Производительность алгоритма оптимизирована до уровня, когда он может быть использован в режиме реального времени. Исходный код с инструкциями по воспроизведению результатов выложен в открытый репозиторий.

Достоверность полученных результатов:

1. Правильность выбора архитектуры системы сбора данных подтверждается широким использованием выбранных подходов, как в коммерческих проектах, так и в проектах с открытым исходным кодом. Действенность используемых инструментов подтверждается их популярностью среди разработчиков программного обеспечения. Эффективность распределенной системы сбора данных была продемонстрирована при горячей замене отдельных подсистем и отдельных компьютеров. Такую замену практически невозможно осуществить в традиционной монолитной архитектуре.
2. Эффективность алгоритмов разделения наложений наглядно продемонстрирована в диссертации как при помощи всестороннего тестирования на сгенерированных данных, так и при помощи амплитудного и временного анализа результатов обработки реальных данных.

Апробация работы. Результаты диссертации докладывались на VII и VIII межинститутских молодежных конференциях «Физика элементарных частиц и космология», международной конференции «The XXI International Scientific Conference of Young Scientists and Specialists» и ряде научных семинаров в ИЯИ и KIT, Карлсруэ.

Личный вклад

Разработка и тестирование всех программных компонентов, а также подготовка к публикации материалов по теме алгоритмов разделения наложенных событий проводились непосредственно автором. Также автор принял активное участие в шести сеансах работы «Троицк ню-масс», по итогам которых были получены результаты мирового уровня:

- наилучшее ограничение на примесь стерильного нейтрино с массой в районе 1 кэВ [3],
- исследование и набор физических данных с помощью детектора нового поколения TRISTAN [12].

Содержание диссертации и основные положения, выносимые на защиту, отражают персональный вклад автора в работу.

Публикации. Основные результаты по теме диссертации изложены в шести печатных изданиях, рекомендованных ВАК. Ссылки на соответствующие статьи указаны в списке литературы[3][5][12][13][14][15].

Объем и структура работы. Диссертация состоит из введения, трёх глав, заключения и двух приложений. Полный объём диссертации составляет 125 страниц, включая 49 рисунков и 13 таблиц. Список литературы содержит 56 наименований.

Глава 1. Эксперименты по поиску стерильных нейтрино

Как уже упоминалось, параметры гипотетического стерильного нейтрино слабо ограничены с точки зрения теории, но максимальный интерес представляют три области масс:

- Легкие стерильные нейтрино (масса порядка одного эВ). Такие легкие нейтрино могут объяснить ряд аномалий, наблюдаемых в экспериментальных данных (например результаты эксперимента LSND[16]). В настоящее время, поиск таких нейтрино ведется в основном в осцилляционных экспериментах с короткой базой. Более подробный обзор экспериментов и соответствующей теории представлен в [17].
- «Средние» нейтрино с массами порядка нескольких кэВ [4] представляют существенный интерес с точки зрения космологии, поскольку могут объяснить загадку темной материи. Именно в этом диапазоне ведет поиск эксперимент «Троицк ню-масс» и о нем пойдет речь в этой работе.
- Тяжелые нейтрино с массами свыше десятков МэВ интересны, поскольку могут позволить решить проблему генерации масс нейтрино в стандартной модели при помощи see-saw механизма [18] и аналогичных теорий.

1.1 Исследование спектра бета-распада трития

Исторически, наиболее точным прямым методом поиска массы электронного нейтрино был исследование бета-спектра трития. Этот же процесс можно использовать и для поиска стерильных нейтрино средних масс. Небольшая примесь стерильного нейтрино с собственной массой, отличной от собственных масс активных компонент, при достаточном разрешении, проявится на форме спектра: на определенной энергии, соответствующей массе стерильного нейтрино, энергии кинематического процесса становится достаточно для испускания тяжелого стерильного нейтрино вместе с электронным [19][20][21][22].

Измеряемый спектр является взвешенной суперпозицией соответствующих собственной массе для каждого аромата нейтрино спектров с весом,

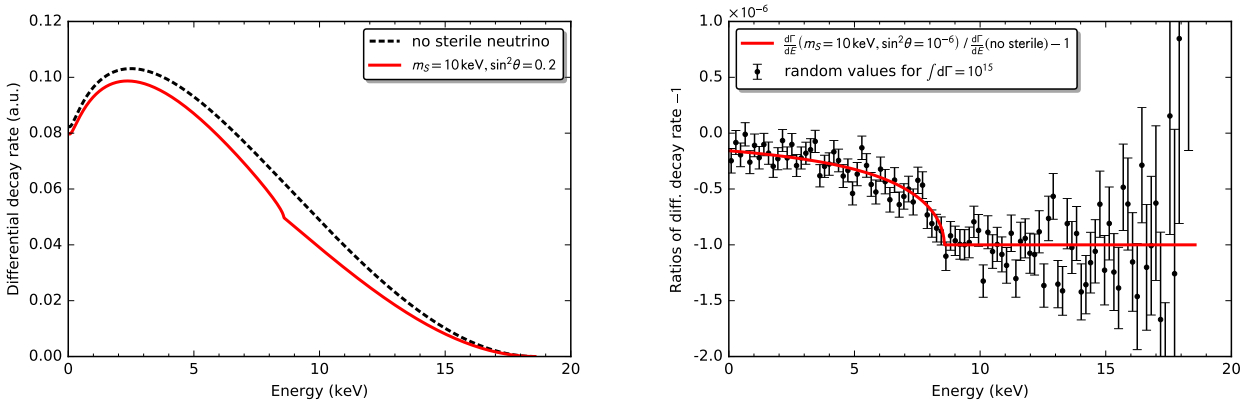

Рисунок 1.1 — a: Сравнение спектров β-распада трития без смешивания (черная пунктирная линия) со смешанным со стерильным нейтрино массой 10 кэВ и углом смешивания $\sin^2\theta = 0.2$ (красная линия). На графике легко заметить сигнатуру в виде перегиба в точке $E = E_0 - m_s$ искажение формы перед перегибом. b: Сравнение спектров β-распада трития без смешивания (черная пунктирная линия) со смешанным со стерильным нейтрино массой 10 кэВ и углом смешивания $\sin^2\theta = 10^{-7}$. Ошибки советуют симуляции $\sim 10^{18}$ электронов.

равным амплитуде перехода из текущего аромата нейтрино в электронный (элемент U_{ei} в матрице смешиваний; здесь i - текущий аромат). Т.к. различие масс между электронным, мюонным и тау-нейтрино слишком мало, чтобы быть различимым в каком либо из действующих экспериментов, вместо суперпозиции состояний используется спектр, соответствующий одной эффективной легкой массе нейтрино, соответствующей формуле 1.1.

$$m(\nu_e)^2 = \sum_{i=1}^{3} \mid U_{ei} \mid^2 m(\nu_i)^2 \tag{1.1}$$

В случае, если электронный нейтрино имеет примесь собственной массы m_s порядка $O(1)$ кэВ, суммарный спектр будет состоять суммы спектра эффективной легкой массы нейтрино и спектра m_s и иметь дифференциальную форму, описывающуюся формулой 1.2.

$$\frac{d}{d\Gamma} = cos^2(\theta)\frac{d}{d\Gamma}(m(\nu_e)) + sin^2(\theta)\frac{d}{d\Gamma}(m_s) \tag{1.2}$$

На графиках 1.1 показаны качественные примеры идеальных формы спектра, на которых легко заметить сигнатуры стерильных нейтрино.

Исследование бета распада трития для поиска стерильных нейтрино с массой порядка кэВ имеет очевидные преимущества: процесс распада имеет сверх-разрешенный тип, поэтому форма спектра описывается точной теоретической формулой. Также тритий имеет относительно небольшое время полураспада и может, при небольшом объеме источника, обеспечить высокую скорость счета (небольшой объем также минимизирует систематические эффекты, связанные с источником). Наконец, тритий имеет граничную энергию бета спектра $E_0 = 18.575$ кэВ, благодаря которой можно проводить поиски в интересном для астрофизиков диапазоне масс.

Анализ спектра бета-распада трития используется в экспериментах:

– «Троицк ню-масс» - в качестве источника используется тритиевый газ; набор спектра осуществляется в интегральном режиме с помощью большого цилиндрического спектрометра, который останавливает частицы по порогу энергии; установка будет подробно описана в работе.
– «KATRIN»[11][23] - установка по принципу работы и структуре схожа с «Троицк ню-масс», однако имеет значительно больший масштаб. Увеличенные размеры позволяют достигнуть разрешающей способности в 200 мэВ.
– «Project 8»[24][25], «Ptolemny»[26] - эксперименты исследуют излучение бета электронов в циклотроне, используется источник атомного трития.

1.2 Электронный захват

Еще одним перспективным способом для поиска является исследование калориметрически измеренного спектра электронного захвата ^{163}Ho . Выбор вещества обусловлен его низкой энергией, доступной для распада равной $Q_{EC} = 2.833 \pm 0.030_{stat} \pm 0.015_{syst}$ кэВ[27][28], благодаря которой на границах энергетического спектра сохраняется достаточная скорость счета и хорошая статистика. Как следствие, в задаче определении массы нейтрино, подход может добиться такой же точности, что и большой эксперимент «KATRIN».

На данный момент есть три больших коллаборации, использующие калориметрический спектр электронного захвата ^{163}Ho и ориентированные на набор статистики большого размера и высокой точности: «Electron Capture in ^{163}Ho

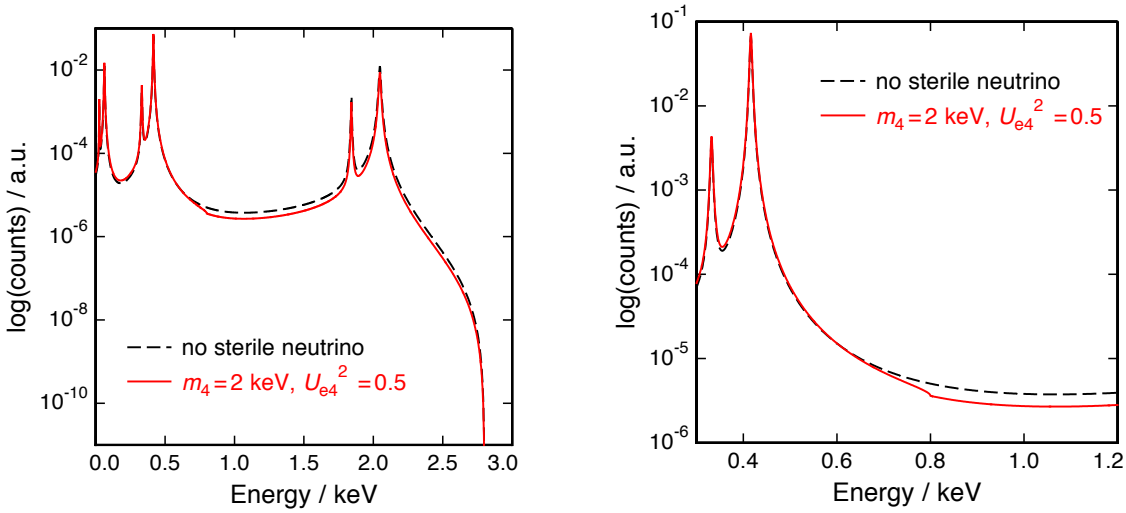

Рисунок 1.2 — a: Сравнение калориметрических спектров ^{163}Ho без воздействия стерильных нейтрино (черная пунктирная линия) и с воздействием тяжелых стерильных нейтрино с массой $m_4 = 2$ кэВ и коэффициентом смешивания $U_{e4}^2 = 0.5$. b: Приближение (a) в область перегиба.

(ECHo)»[29], «Electron Capture Decay of ^{163}Ho to Measure the Electron Neutrino Mass with sub-eV sensitivity (HOLMES)»[30], «Neutrino Mass via ^{163}Ho Electron Capture Spectroscopy (NuMECS)»[31]. Сейчас коллаборации нацелены на измерения массы нейтрино, однако, т.к. при калориметрических измерениях каждое событие создает измеримый сигнал и может быть записано, установки способны набрать спектр во всем диапазоне и провести поиск стерильного нейтрино.

Сигнатура стерильных нейтрино, как и в тритиевом бета спектре, проявляется на спектре в виде небольшого перегиба. Подробнее про модель спектра можно прочитать в [4]. По спектру ^{163}Ho можно определить существование стерильных нейтрино в диапазоне до $Q_{EC} = 2.5$ кэВ. На графиках 1.2 показано сравнение модельных форм спектров.

1.3 Прямое измерение нейтрино

В данном способе измеряются свойства реликтовых стерильных нейтрино. Подход имеет свои особенности. Из положительных можно отметить, что такие измерения помимо поиска стерильных нейтрино также исследуют приро-

ду и свойства темной материи. Т.о. эксперимент, в случае соответствия модели, станет убедительным доказательством того, что темная материя состоит из стерильных нейтрино, а, в случае несоответствия - данные все равно могут быть использованы для проверки других гипотез о темной материи.

Прямые измерения могут быть реализованы через захват космических электронных нейтрино радиоактивными ядрами, испытывающими бета-распад[32][33]. Посредством смешивания с активным электронным (анти) нейтрино $\nu_e(\bar{\nu}_e)$, кандидат на темную материю N_1 массой порядка $O(1)$ может подвергаться реакции захвата $N_1 + \mathcal{N}(A,Z) \to \mathcal{N}'(A, Z \mp 1) + e^{\pm}$, где A, Z - массовый и атомный номера исходного ядра соответственно. Сигнатуры реакций захвата измеряются через испускаемые в результате моноэнергетические электроны (позитроны), энергия которых находится за границей энергий соответствующего бета распада. Измерение расстояния между процессами распада и захвата напрямую определит наличие стерильных нейтрино с массой порядка кэВ в темной материи и определит соответствующие массу и коэффициенты смешивания.

Глава 2. Установка «Троицк ню-масс»

Установка предназначена для проведения прецизионных измерений бета спектров от трития в диапазоне энергий от 5 кэВ до порядка 35 кэВ. В основе ее работы лежит принцип так называемого адиабатического движения электронов в магнитном поле. Под адиабатичностью движения здесь подразумевается не отсутствие потерь энергии (магнитное поле не совершает работы над заряженной частицей, поэтому потерь быть и не может), а движение электрона вдоль одной силовой линии.

На рисунке 2.2 схематично изображены компоненты «Троицк ню-масс». Установка состоит из: интегрального электростатического спектрометра с магнитной адиабатической коллимацией, безоконного источника электронов с молекулярным тритием в газообразном состоянии, вакуумной системы откачки, криогенной системы, обеспечивающей рабочую температуру сверхпроводящих соленоидов спектрометра и термостабилизацию источника, высоковольтную систему и систему сбора данных.

Безоконный тритиевый источник представляет собой трубу длиной 3 м и диаметром 5 см, помещенную внутрь криостата со сверхпроводящими магнитами, который генерирует внутри источника аксиальное поле с величиной потенциала до 0.8 Тл.

Торец источника со стороны спектрометра соединен с системой магнитной транспортировки, которая перемещает электроны бета-распада газообразного трития в объем спектрометра. Поле внутри системы достигает значений 5 Тл и ее коллимационные магниты расположены под углом друг к другу образуя путь в виде «зигзага», не имеющий сквозного прямого пути. Такая конфигурация магнитов препятствует прохождению незаряженных частиц в спектрометр.

Рисунок 2.1 — Установка «Троицк ню-масс».

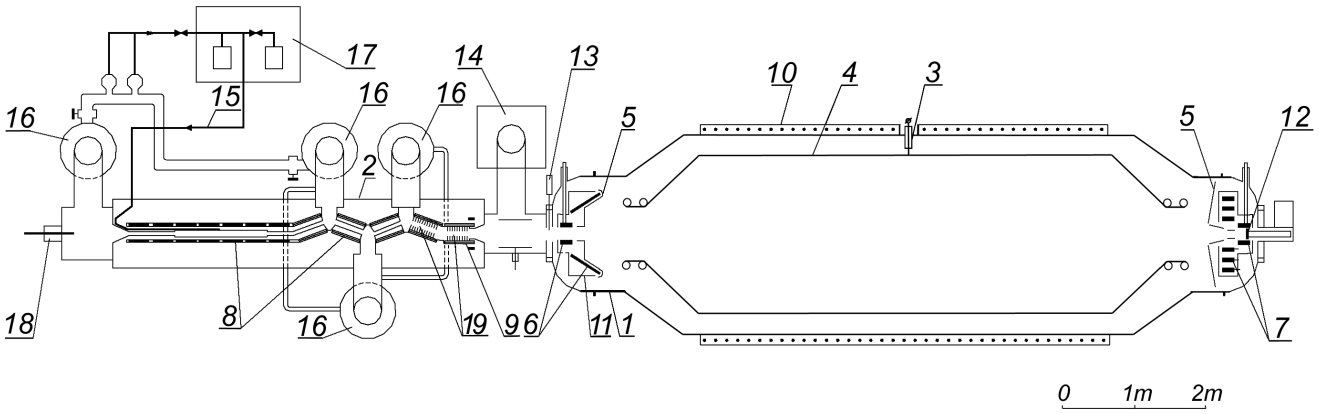

Рисунок 2.2 — Схема установки. Обозначения: 1 - вакуумный объем спектрометра; 2 - вакуумный объем источника; 3 - высоковольтный ввод; 4 - электрод спектрометра; 5 - заземляющие электроды; 6, 7, 8, 9 - сверхпроводящие катушки, 10 - индукционные катушки (не сверхпроводящие), 11 - охлаждающий кожух с жидким азотом; 12 - детектор, охлажденный до температуры жидкого азота; 13 - аварийный шибер, 14 - магниторазрядный насос; 15 - тритиевый контур; 16 - ртутные диффузионные насосы; 17 - система ввода и очистки трития; 18 - электронная пушка; 19 - аргоновая ловушка.

На рисунке 2.3 изображен электростатический спектрометр, используемый на установке.

Спектрометр работает в интегральном режиме. С помощью высоковольтной системы на основной высоковольтный электрод подается запирающее напряжение. При прохождении электрона, в случае, если его энергия превышает выставленное запирающее напряжение - электрон проходит потенциальный барьер и попадает в установленный за ним детектор. Электроны с более низкими энергиями отражаются от потенциала и не доходят до детектора. Проведя серию наборов событий при разных значениях запирающего напряжения мы получим и интегральный спектр энергий рожденных во время бета-распада трития электронов.

Разрешающая способность спектрометра определяется его геометрией и может потенциально достигать долей электронвольта (например в эксперименте «KATRIN», упоминавшемся ранее [11]; «Троицк ню-масс» к сожалению из-за небольших размеров помещения имеет разрешение на один порядок меньше). Использование спектрометра позволяет значительно снизить требования к разрешению детектора, т.к. оно перестает быть определяющим фактором.

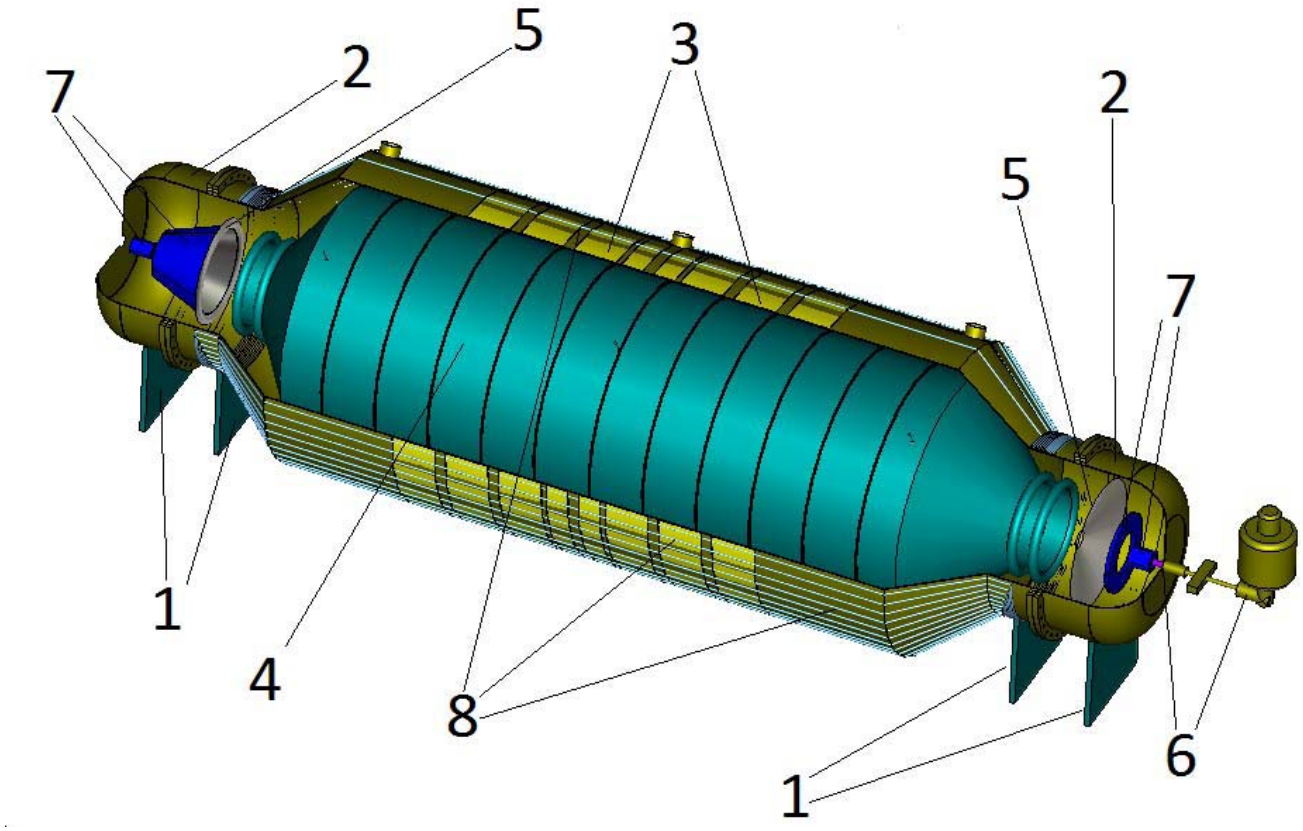

Рисунок 2.3 — Схема спектрометра. Обозначения: 1 - опоры спектрометра, 2 - входная и выходная чашки, 3 - теплые аксиальные обмотки, 4 - основной высоковольтный электрод, 5 - электроды под нулевым потенциалом, 6 - система детектора с охлаждением жидким азотом, 7 - набор сверхпроводящих магнитов.

Амплитуды зарегистрированных событий используются в основном для контроля качества набора, поиска и фильтрации аномальных событий, (в принципе можно использовать детектор и в счетном режиме). Адиабатичность переноса электронов в спектрометре обеспечивается конфигурацией магнитного поля спектрометра, которое формируется с помощью системы сверхпроводящих соленоидов, размещенных на поверхности.

Т.к. в работе будет рассматриваться в основном система управления набором - рассмотрим подробнее элементы системы, делая акцент на особенности управления и способы автоматизации контроля.

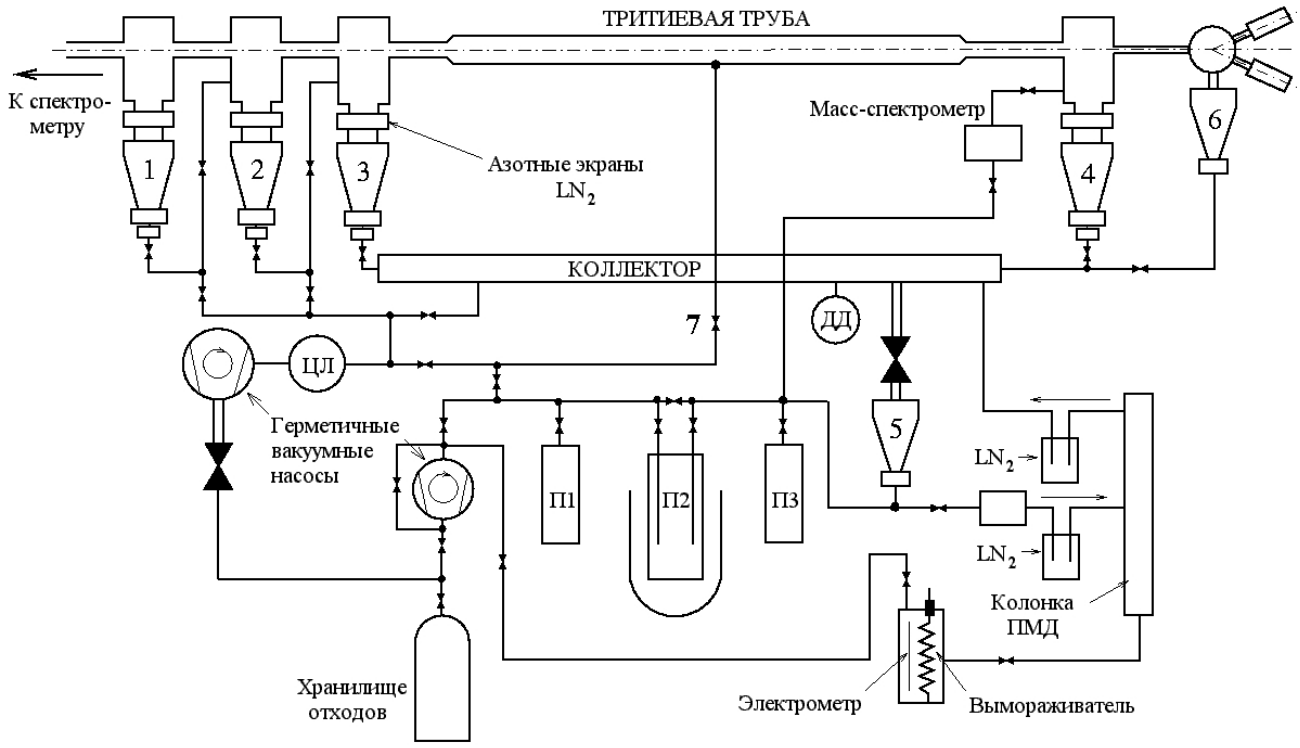

Рисунок 2.4 — Схема тритиевого источника. Обозначения: 1 – ртутный насос «Р4»; 2 – ртутный насос «Р3»; 3 – ртутный насос «Р2»; 4 – ртутный насос «Р1»; 5 – бустерный ртутный насос; 6 – ртутный насос откачки задней секции; 7 – натекатель; ЦЛ – цеолитовая ловушка; ДД – датчик давления; П1 – патрон-хранилище; П2 – очистной патрон; П3 – транспортный контейнер.

2.1 Тритиевый молекулярный источник электронов

На рисунке 2.4 представлена схема тритиевого источника.

Низкая граничная энергия бета-распада трития (18575 эВ) с одной стороны обеспечивает возможность установки достаточного напряжения на спектрометре. С другой стороны такие низкие энергии делают принципиально невозможным создание какого-либо барьера между объемами спектрометра и источника. Поэтому между источником и спектрометром должна находится мощная система перехвата и откачки газа, обеспечивающую снижение концентрации трития по пути от источника к спектрометру практически на 10 порядков - от $5 * 10^{14}\mathbf{см}^{-3}$ в источнике до $10^{5}\mathbf{см}^{-3}$ в спектрометре.На практике выполнение таких требований реализуется системой из пяти ртутных насосов: 3 насоса («Р1», «Р2», «Р3») расположены между тритиевой трубой источника и спектрометра, один - между тритиевой трубой и задней секцией («Р4») и один

аргоновый насос прямо перед спектрометром. К откачному порту насоса «P4» подключен масс-спектрометр MX-7340, используемый для анализа и мониторинга изотопного состава газа в источнике. Насосы «P1», «P2», «P3» объединены в цепь дифференциальной откачки (т.е. в последовательную цепь, где выхлоп насоса соединяется с откачным портом следующего насоса т.о. обеспечивая более высокий предельный вакуум). Откачанный насосами газ попадает в коллектор, где с помощью бустерных насосов прогоняется через систему изотопной очистки, а затем через тонкий натекатель возвращается обратно в тритиевую трубу. Таким образом, осуществляется замкнутый контур циркуляции трития в системе.

Помимо мощной системы откачки источник также требует термостабилизации трития: т.к. он находится при температурах порядка 30 K даже небольшие колебания температуры способны значительно изменить концентрацию газа и соответственно повлиять скорость счета набираемых данных (флуктуации в 1 K меняют концентрацию трития более чем на 3 %).

Система источника управляется вручную. В процессе набора данных проводится автоматизированный мониторинг давлений в насосах, температуры и изотопного состава источника.

2.2 Спектрометр

Спектрометр представляет собой цилиндр диаметром 2.6 м и длиной 8 м, внутри которого на электроизолирующих подвесах установлен цилиндрический подключенный к высоковольтной стойке электростатический электрод, создающий основной запирающий потенциал. Внешний цилиндр с торцов закрывается полусферическими «чашками», внутри которых располагаются сверхпроводящие магниты: пинч-магнит со стороны источника и магнит детектора. Эти магниты корректируют поле основного электрода делая из него магнитную «бутылку» в спектрометре (качественно форма изображена на рисунке 2.5).

Для обеспечения плавности изменения диаметра трубки тока по мере прохождения спектрометра в чашках также находятся согласовывающие магниты. На поверхности спектрометра расположены еще три электромагнитные системы в виде продольных и аксиальной обмоток, образующих сетку. Они

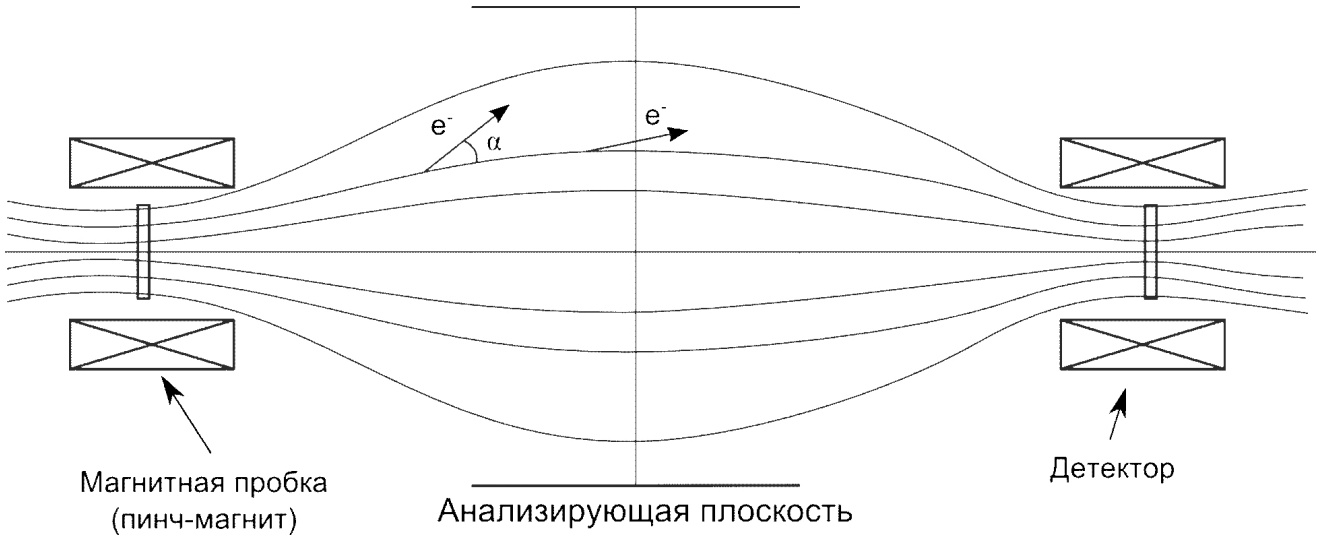

Рисунок 2.5 — Конфигурация магнитного поля внутри спектрометра.

используются для дополнительной коррекции магнитного поля в центре спектрометра - две из них компенсируют магнитное поле земли и другие внешние поля, третья аксиальная позволяет сужать трубку и держать силовые линии внутри объема электрода.

Во время набора, спектрометр управляется автоматизировано с помощью задания напряжения, подающегося на электрод, выполняемого аппаратной частью высоковольтной системы. Управление токами формирующих магнитов также автоматизированно.

2.3 Криогенная система

Криогенная система установки «Троицк ню-масс» осуществляет поддержание необходимой для обеспечения сверхпроводимости температуры на соленоидах спектрометра и термостабилизацию трубки источника. Схема криогенной системы показана на рисунке 2.6.

Охлаждение криостатов осуществляется через циркуляцию по ним жидкого гелия, который проходит последовательно от криостатов трубки и соленоидов источника через криостаты спектрометра в ожижитель TCF-50. Для снижения теплопотерь и уменьшения нагрузки на рефрижератор, гелиопроводные трубы помещаются в экраны наполненные жидким азотом. Азотные экраны имеют свою, независимую от гелия, систему циркуляции. Помимо это-

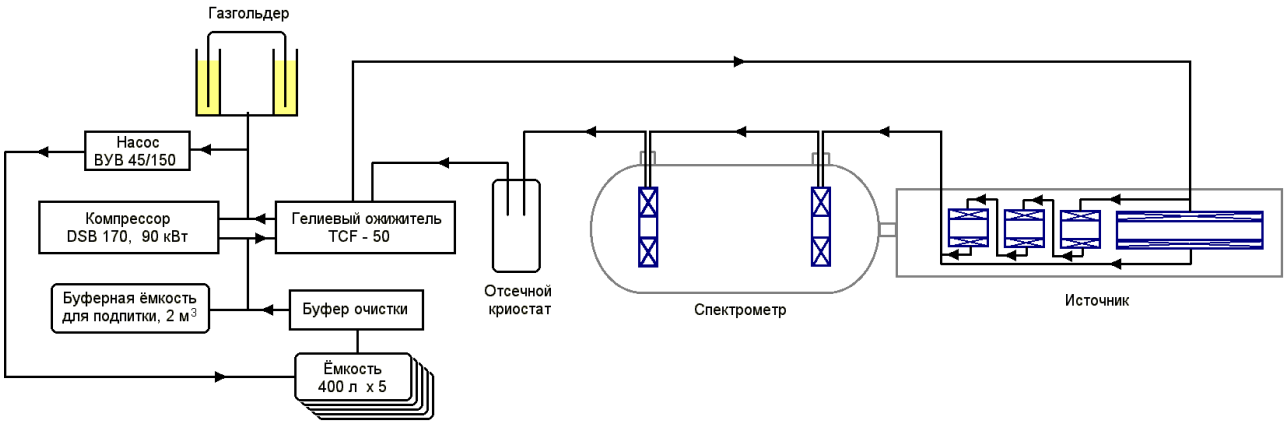

Рисунок 2.6 — Схема криогенной системы.

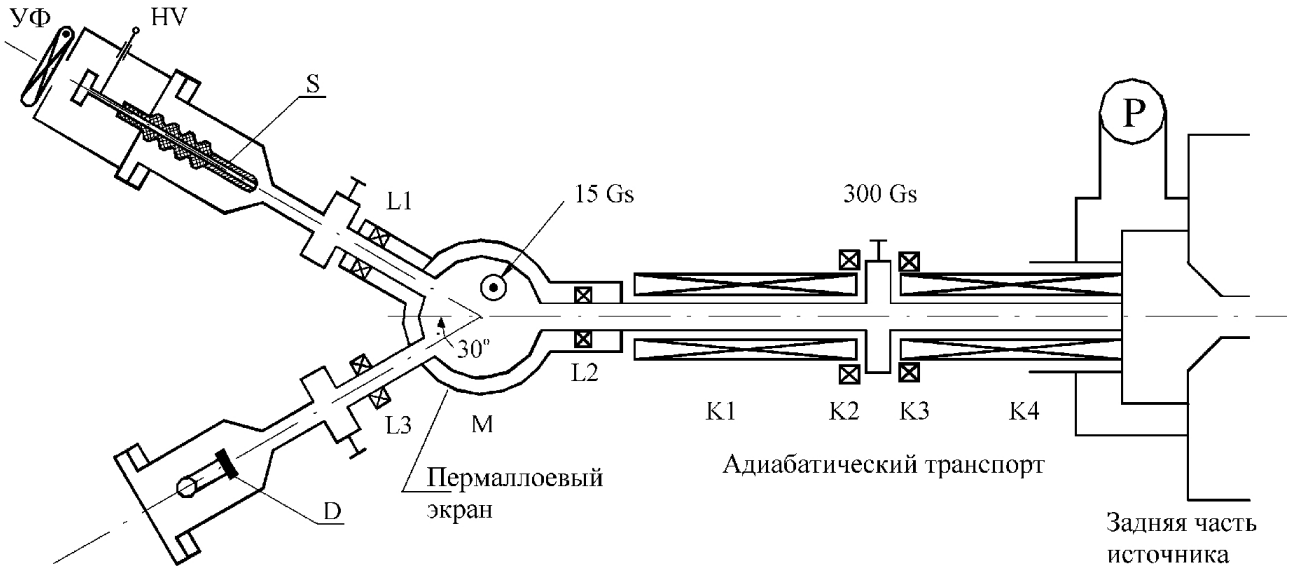

Рисунок 2.7 — Схема электронной пушки.

го с помощью отдельных дьюаров на установке осуществляется охлаждение детектора, электронной пушки и откачивающих и бустерных насосов. Более подробная схема работы криогенной системы описана в [34][35].

2.4 Электронная пушка

Схема электронной пушки, использующейся на установке «Троицк нюмасс» изображена на рисунке 2.7.

Принцип работы основан на выбивании электронов с золотой поверхности с помощью ультрафиолета и последующего их ускорения в электростатическом

поле до нужных энергий. В пушке предусмотрена фильтрация электронов от фотонов УФ лампы и ионов источника исполненная в виде поворотного магнита, расположенного после ускоряющего конденсатора. Ускоряемые в сторону поверхности с золотым напылением ионы при столкновении с ней производят дополнительные электроны тем самым увеличивая эффективность пушки.

После поворотного магнита по пути следования электронов расположены коллимационные магнитные линзы обеспечивающие сохранение углового распределения электронов. После настройки поворотного и коллимационных магнитов пушки выходное распределение электронов изотропно и имеет максимальный угол вылета = 0.0513 рад. Подробнее о строении и калибровке пушки написано в [36].

2.5 Сбор данных

На рисунке 2.8 показана схема исходной системы сбора данных установки «Троицк ню-масс», использовавшаяся до 2010 года. Базовыми блоками являются группы аппаратных модулей, из которых каждая контролирует отдельную часть установки. Группы разделяются по крейтам, которые при помощи контроллера крейта «КК» (аналог «POLON 106») и интерфейсной платы «PC-UNIBUS» подключаются к управляющему компьютеру. Управление набором производится программой, выполненной в монолитной архитектуре, которая с помощью подключенных контроллеров управляет модулями в крейтах. Система состоит из двух независимых компонент:

1. Управление высоким напряжением и запись данных детектора.
2. Система контроля температур криостатов и считывание данных медленного контроля.

Подробнее, про исходную архитектуру системы сбора можно прочитать в [36].

Далее в работе мы будем касаться только компоненты управления высоким напряжением (крейт HV и подключенная к нему аппаратная составляющая) и системы регистрации событий на детекторе (крейты A2, ADC). С точки зрения этих компонент, типичный сценарий набора данных на установке состоит из последовательных наборов событий с детектора за фиксированный отрезок времени при фиксированном напряжении спектрометра (точек). Точ-

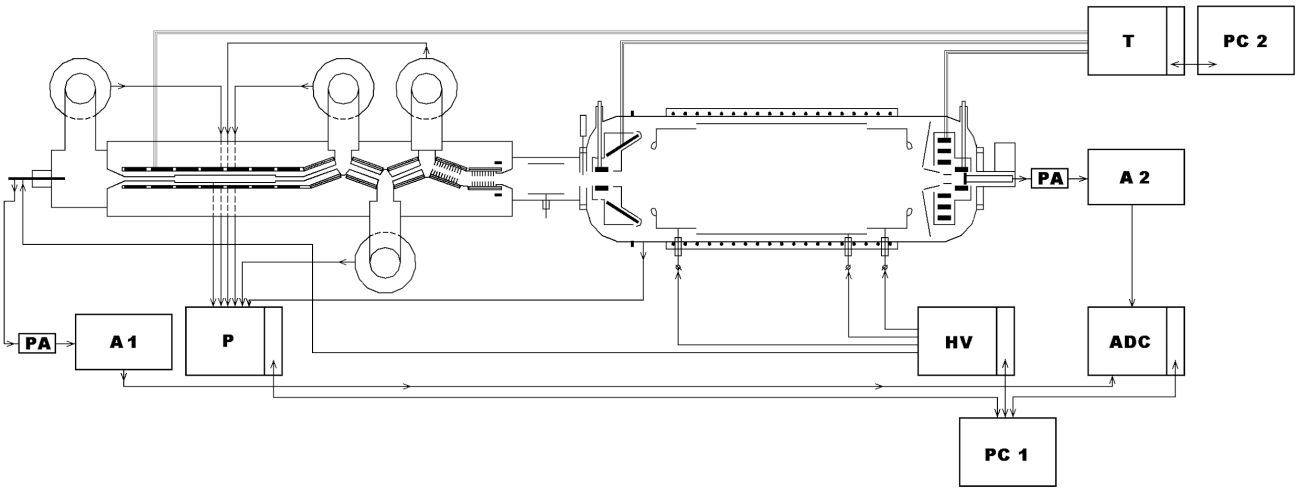

Рисунок 2.8 — Схема исходной системы сбора данных. Обозначения: PA – предварительные усилители, A1, A2 – крейты аналоговой обработки сигнала (NIM), ADC – крейт оцифровки сигнала, HV – высоковольтная система, P – крейт оцифровки давлений, T – крейт измерения температур, PC 1, PC 2 – персональные компьютеры.

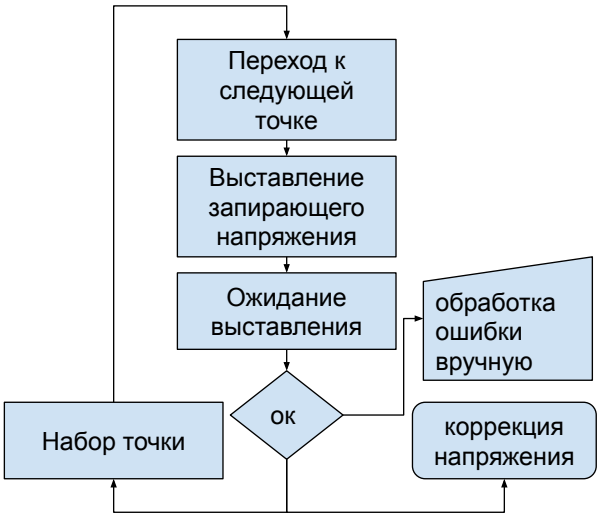

Рисунок 2.9 — Блок-схема набора по сценарию.

ки отличаются друг от друга величиной запирающего напряжения и, в случае большой разницы в скоростях счета, временем набора. На рис 2.9 показана блок схема цикла набора.

Помимо активного управления во время набора измерений производится непрерывная запись и мониторинг данных медленного контроля, таких как давления ртутных насосов, температуры элементов криогенной части системы и значения магнитных полей.

2.6 Аппаратная составляющая системы сбора данных

Созданная в процессе работы система сбора данных установки «Троицк ню-масс» аппаратно опирается на готовые наработки, использующиеся на установке с 1993 года. Конкретно, система использует электронику установленную в детекторной части установки и электронику, отвечающую за работу высоковольтной стойки. Для упрощения понимания работы модернизированной системы сбора данных, которая будет изложена в последующих главах, опишем кратко устройство и принцип работы этих двух компонент установки.

2.6.1 Система регистраций событий детектора

При создании аппаратного комплекса, отвечающего за набор регистрируемых детектором событий, которая впоследствии стала использоваться в эксперименте «Троицк ню-масс» закладывалась необходимость регистрации событий в большом диапазоне скоростей счета, границы которого отличаются на шесть порядков - от нескольких килогерц до единиц милигерц. При этом, т.к. предполагалось измерять среднюю скорость счета регистрируемых событий - дополнительным требованием для создаваемой системы стала необходимость исключения пропусков импульсов при наборе.

Установка «Троицк ню-масс» использует в качестве основного регистрирующего детектора изготовленный в ПИЯФ по специальному заказу плоский Si(Li) детектор с диаметром чувствительной области 17 мм, ёмкостью порядка 15 пФ со слоем золота 20 мкг/см2, охлаждаемый до температуры жидкого азота. Приходящий на детектор сигнал импульса попадает в предварительный усилитель ПУМА-73, головной каскад которого расположен рядом с детектором и тоже охлаждается; остальная часть предусилителя, в целях упрощения эксплуатации без нарушения вакуума, вынесена на 1 м от детектора. Из предварительного усилителя сигнал попадает на основной усилитель ORTEC 572, обрабатывающий импульсы а методом однократного дифференцирования и интегрирования с одинаковыми постоянными и имеющим время формирования - 0.5 мкс. Далее идет оцифровка сигнала, которая осуществляется 12-разряд-

ным спектрометрическим аналого-цифровым преобразователем типа ADC-4K производства ПИЯФ. АЦП работает по принципу поразрядного взвешивания с динамическим выравниванием и имеет постоянное время преобразования составляющее 3.91 мкс. Для сохранения параметров регистрируемых импульсов применяется специальный блок MADC (модуль буферной памяти с фиксацией времени прихода каждого события), разработанный и изготовленный по специальному заказу в ЛВЭ ОИЯИ (г. Дубна), и выполненный в стандарте КАМАК. MADC имеет шаг оцифровки времени от 50 до 200 нс в зависимости от длительности набора событий. Емкость модуля составляет 1.5 Мегабайта, которой хватает на запись 256K событий - при записи каждое событие занимает 48 байт: 12 на амплитуду, 28 на время прихода и 8 - на служебную информацию. Взаимодействие между АЦП и MADC можно описать следующей последовательностью действий:

- При превышении входным сигналом установленного порога срабатывает формирователь, открывающий вход АЦП. Вход открывается на фиксированный отрезок времени соответствующий времени формирования сигнала с небольшим запасом, при котором попадающий сигнал будет заведомо содержать пик импульса. В дополнение срабатывание формирователя также приводит к инкрементации показателя счетчика 4СчБ.
- После открытия входа АЦП начинается непрерывный поиск пика импульса, после нахождения которого начинается преобразование.
- В момент начала преобразования АЦП подаёт сигнал в MADC, который записывает в текущее событие метку времени и ожидает завершение преобразования. В процессе преобразования сигнала вход АЦП автоматически блокируется.
- После завершения преобразования, которое занимает 3.91 мкс, АЦП выставляет полученный код на внешнюю шину (4) и подает сигнал строба. MADC по стробу считывает выставленный код и записывает его ко времени, которое было установлено на предыдущем шаге. После, MADC выдает сигнал подтверждения для АЦП. Запись амплитуды события в память суммарно занимает 100 нс.
- Система готова к приему и обработке следующего события.

Описанная схема набора изображена на временной диаграмме на рисунке 2.10; на рис 2.11 приведена полная схема аппаратной оцифровки.

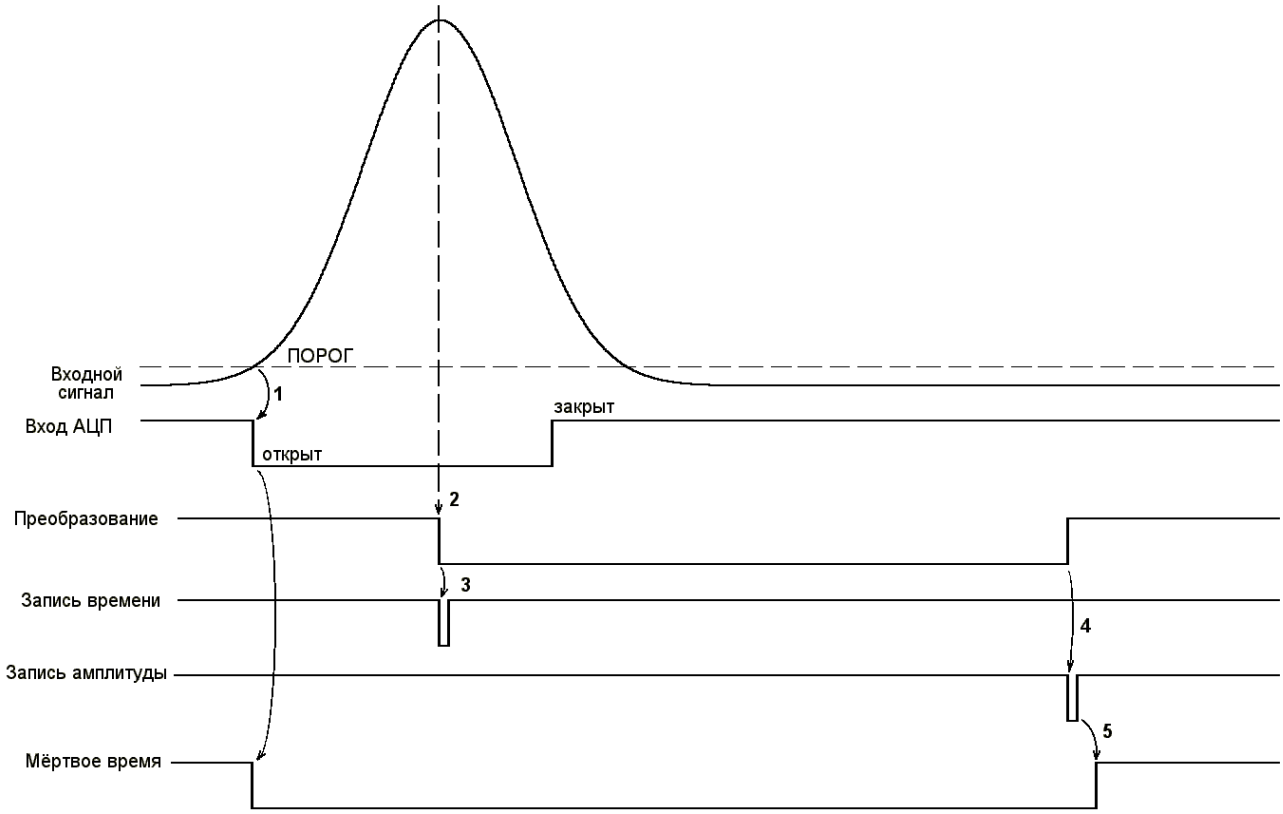

Рисунок 2.10 — Оцифровка и запись события в MADC.

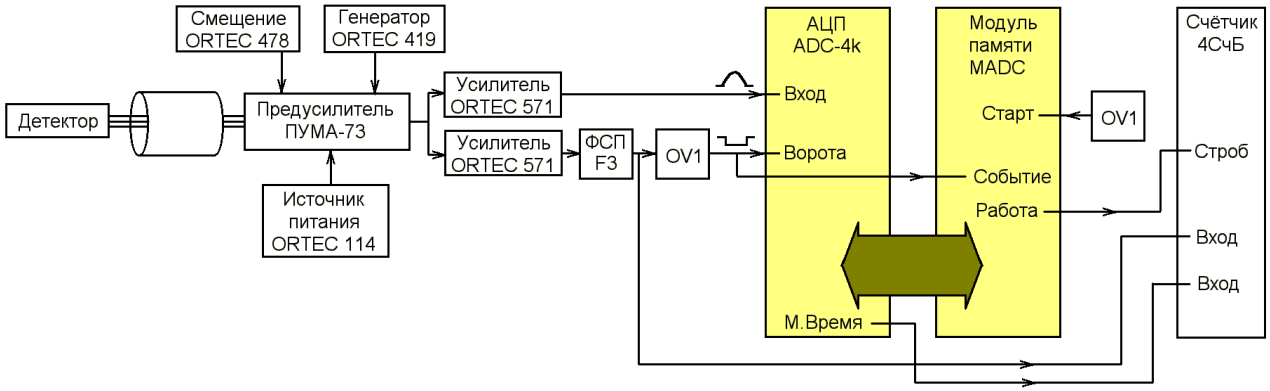

Рисунок 2.11 — Аппаратная обработка сигнала детектора.

Мёртвое время системы сбора отсчитывается от момента открытия входа АЦП до конца цикла записи в память MADC и составляет 6.0 мкс.

Для обхода аппаратных ограничений на пропускную способность транспортной шины КАМАК, АЦП и MADC соединены между собой внешней шиной. Такая конфигурация позволяет добиться быстродействия, недостижимого ни на одном из индустриальных стандартов, используемых во время проектирования системы.

2.6.2 Система управления запирающим напряжением

Аппаратный комплекс, осуществляющий управление и контроль напряжения спектрометра вынесен в отдельную стойку, находящуюся в основании спектрометра. Основными элементами высоковольтной системы являются:

- Прецизионный цифроаналоговый преобразователь В1-13 (прибор для проверки вольтметров); в системе сбора данных этот прибор используется для установки опорного напряжения всей системы стабилизации. С помощью изменения опорного напряжения происходит задания величины задерживающего потенциала спектрометра. Управление устройством осуществляется через установленный в стойку КАМАК с помощью 24-разрядного параллельного регистра типа POLON 350. Один канал модуля осуществляет запись в ЦАП управляющих команд, другой задает величину напряжения
- Управляемый линейный высоковольтный преобразователь HV-30, изготовленный по заказу в Обнинске. Преобразователь имеет коэффициент увеличения входного напряжения порядка 5500 раз. Помимо аналогового входа, через который с помощью ЦАП В1-13 задается опорное напряжения, устройство имеет дополнительный разъем с интерфейсом взаимодействия RS-485, с помощью которого можно проводить тонкую настройку напряжения в пределах 1 кВ.
- Делители напряжения с коэффициентом 1:2026. Применяются для считывания установленного преобразователем HV-30 напряжения с целью его контроля.

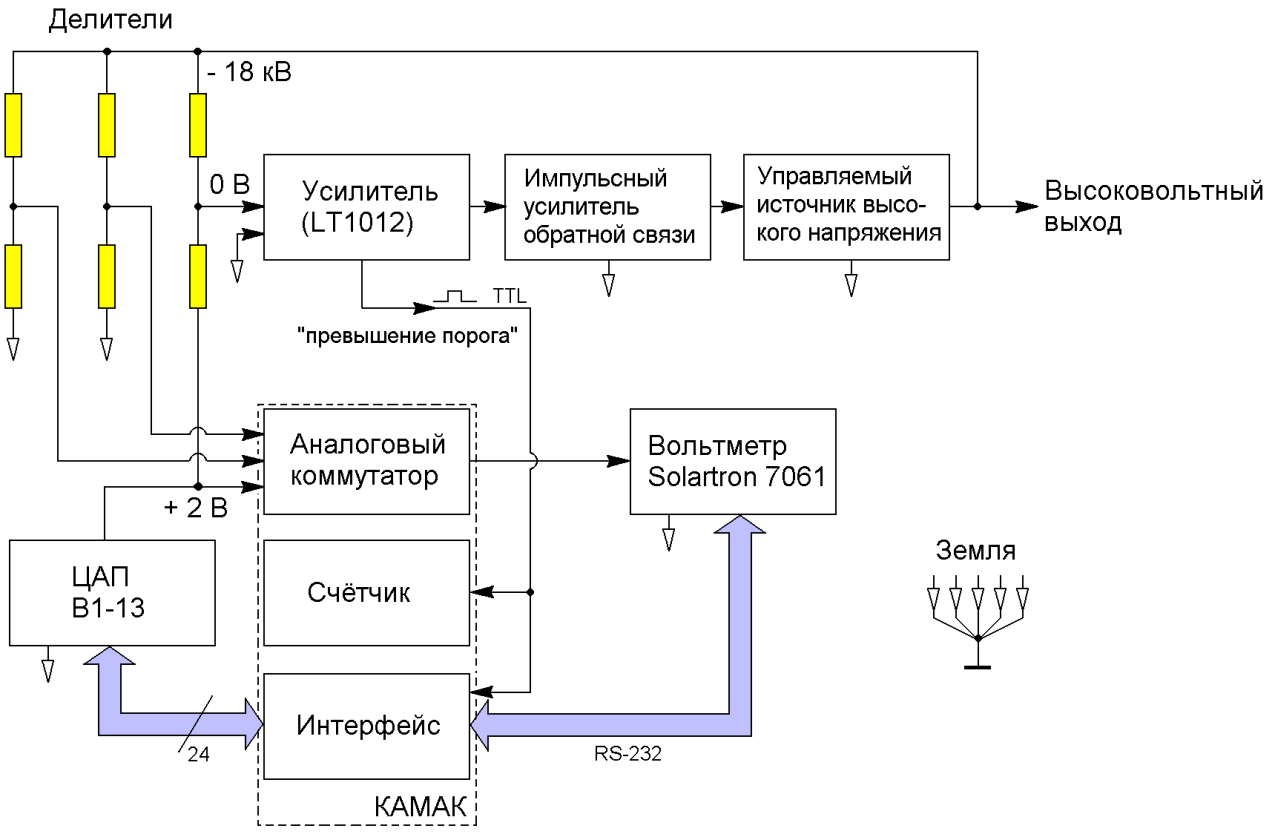

Рисунок 2.12 — Схема работы высоковольтной стойки.

– Прецизионные вольтметры Solarton 7061, которые с помощью аналогового измерительного коммутатора 164 соединяются с делителями и считывают с них напряжение. Вольтметры имеют возможность программного управления через последовательный интерфейс RS-232.
– Стабилизирующий модуль, принцип действия которого будет описан ниже.

Схема работы высоковольтной стойке изображена на рисунке 2.12. Источниками питания для системы являются магистраль крейта КАМАК и два дополнительных источника типа ТЕС-14, которые управляются с помощью ЦАП В1-13 и встроенный в блоки интерфейс RS-232.

Схему кратковременной аппаратной стабилизации высоковольтного напряжения можно представить в виде следующей последовательности действий:

1. Установленное высоковольтное напряжение (отрицательное) и опорное напряжение с ЦАПа (положительное) подаются на два плеча делителя. На средней точке делителя должен установиться нулевой потенциал, если эти два напряжения соответствуют друг другу.
2. Потенциал средней точки делителя усиливается операционным усилителем с очень малым входным током (LT 1012). Тут же стоят и

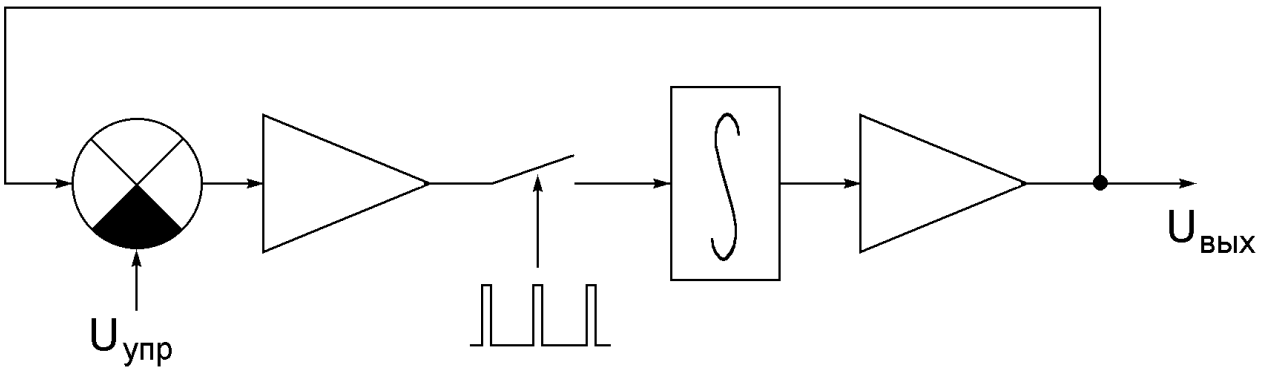

Рисунок 2.13 — Схема стабилизатора высокого напряжения.

компараторы, следящие за отклонениями высоковольтного напряжения («быстрая» схема сравнения).

3. Усиленный сигнал подаётся на вход интегрирующего импульсного усилителя обратной связи. Вход этого усилителя открывается на малое время и сигнал рассогласования попадает на интегратор, увеличивая либо уменьшая управляющее напряжение на небольшую величину. Таким образом удаётся избежать «раскачки» довольно инерционной системы управления высоковольтным напряжением и сделать ее нечувствительной к кратковременным помехам и пробоям (см. рис. 2.13).
4. Управляющий сигнал попадает на вход линейного высоковольтного преобразователя напряжения.

Описанная схема позволяет устранить кратковременные скачки в системе, однако при долгосрочных измерениях (для времен измерений порядка 10 часов и более) выставленное напряжение все равно будет «уплывать» от заданного. Устранение подобных отклонений в высоковольтной стойке ложится на алгоритмы управляющего компьютера. Аппаратно, стойка позволяет считывать напряжение с делителей с помощью вольтметров и задавать через ЦАП и RS-232 интерфейс преобразователя напряжения. Это позволяет при постоянном мониторинге показаний вольтметров вычислять отклонения от желаемых значений и вносить их в устанавливаемое напряжение. Т.к. преобразователь имеет 2 входа для установки напряжения, один их них (аналоговый) используется для установки полного значения а другой (RS-232) для внесения коррекций, обеспечивающих долгосрочную стабильность.

Глава 3. Архитектура системы сбора

3.1 Формат хранения и передачи данных

В процессе проектирования новой архитектуры системы сбора установки «Троицк ню-масс» и с учетом предыдущего негативного опыта потери данных вследствие ненадежности или неприспособленности формата хранения (например разбиения на несколько файлов логически атомарной единицы данных, потеря описания формата данных, изменения формата в процессе набора и др) было решено выработать единый формат хранения и передачи данных и использовать его в новой системе.

Для формата были сформулированы следующие требования:

1. Формат должен быть максимально гибким. Его использование потенциально не должно ограничиваться только установкой «Троицк ню-масс» - должна быть возможность имплементации на другие установки и фреймворки.
2. Передаваемые и сохраняемые на жесткий диск пакеты данных должны иметь одинаковый формат. Различие форматов ведет к ненужному усложнению для понимания работы системы в то же время не создавая значительного преимущества в производительности.
3. Пакет данных должен иметь возможность содержания текстовых полей (метаданных) и бинарных полей (набранные данные). Оба вида данных в пакете опциональны. Типичная логически цельная единица данных для экспериментальной физики частиц практически всегда представляет из себя метаданные, содержащие информацию о параметрах набора и собственно сами набранные данные, которые чаще всего целесообразно хранить в бинарном виде ввиду их объема. С учетом этого описанное требование выглядит естественным для специфики данных. Объединения двух видов данных в один неделимый пакет усложняет потерю части набранных данных при дальнейших манипуляциях с ними.
4. Метаданные пакета должны читаться без необходимости прочтения всего пакета. Выполнение этого требования обеспечивает возможность

быстрого предварительного анализа, фильтрации и индексации набранных данных.

5. Содержание метаданных пакета должно иметь человекочитаемый формат. Содержание пакета должно быть понятно при работе через текстовый редактор. В этом случае пакет можно будет открыть без дополнительного ПО и в дальнейшем понять, какое ПО необходимо для полноценной обработки. Также облегчается отладка и понимание работы системы.
6. Протоколом передачи сообщений должен служить TCP/IP. Это обеспечит относительно простое встраивание в установку в силу доступности и дешевизны соответствующего оборудования и возможности протягивания кабелей на длинные дистанции. Современное оборудование позволяет построить сеть с пропускной способностью до 100 Гбит/с, что покрывает практически все потенциальные требования для экспериментов. С другой стороны наличие огромного числа имплементаций библиотек для работы с протоколом практически для всех языков обеспечивает простоту разработки модулей сбора данных.
7. Хранение данных на жестком диске должно реализовываться средствами файловой системы и быть имплементировано в виде древовидной структуры, состоящей из папок и файлов. Для обеспечения чистоты и объективности результатов анализа, необходимо всегда опираться только на исходные данные, полученные напрямую с установки. Поэтому собранные и сохраненные в процессе эксперимента пакеты данных должны быть принципиально неизменяемыми. В случае, если для данных требуется коррекция (например перекалибровка для отдельных датасетов) она должна быть выполнена в виде отдельного файла, задающего мутацию данных. Такая специфика делает хранение данных в привычных форматах таких как реляционные базы данных не оптимальным ввиду усложнения работы с ними и неиспользования их сильных сторон из-за иммутабельности. Хранение в виде древовидной структуры в файловой системе достаточно и обеспечивает наглядность при работе с данными.

На момент разработки из известных и реализованных форматов, под требования частично подходил только формат Http-multipart, однако от его использования было решено отказаться, т.к. сам формат не смотря на продол-

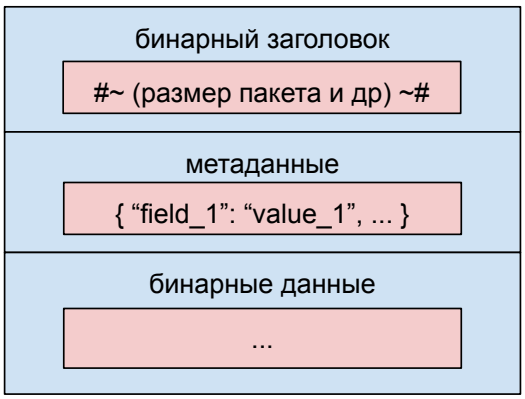

Рисунок 3.1 — Стркутура пакета DataForge Envelope.

жительное существование не обрел популярности. В частности для него нет работающего парсера на Python, способного десериализовать пакеты, содержащие бинарные поля (однако есть временное решение в виде коммита[37]). Кроме того для хранения пакета используются несколько файлов, что затрудняет процесс обработки и хранения данных а передача пакетов требует использования http, который в общем случае не предназначен для передачи бинарных данных и для нашей задачи является излишней настройкой на протокол TCP/IP.

Ввиду отсутствия на момент разработки готовых форматов, удовлетворяющих всем сформулированным требованиям было принято решение о разработке собственного формата для системы хранения. При этом, для обеспечения долговечности и упрощения разработки, реализации все равно максимально использовали уже существующие форматы и их реализации.

3.1.1 Формат DataForge Envelope

В качестве формата хранения и передачи данных используется формат dataforge-envelope из фреймворка DataForge[38]. Фреймворк DataForge позволяет обрабатывать данные в области экспериментальной физики частиц. В основе архитектуры лежит data-driven принцип - любая обработка представляется как граф математических преобразований исходных данных. Каждый узел графа содержит правила обработки, в случае успешной обработки - кэш данных с результатом. Все данные фреймворка хранятся в формате dataforge-envelope. Рассмотрим подробнее этот формат.

На рисунке 3.1 схематично изображена структура пакета. В использующемся протоколе TCP/IP передача данных происходит через бинарный поток. При этом фактически данные передаются фрагментированными пакетами. Фрагментация обуславливается не логической связностью передаваемых данных а передаваемым объемом и общим состоянием сети. TCP/IP не имеет в себе функционала логического объединения данных а обеспечивает только правильность последовательности прихода пакетов. Поэтому при передачи пакетов больших объемов на модуль обработки ложится задача выделения и склеивания данных из общего потока. Существует 2 способа объединения данных:

1. Объединение с помощью открывающих и замыкающих специальных последовательностей символов. В этом случае перед началом пакета вписывается заголовок определяющий начало пакета далее записывается уникальная последовательность бинарных символов определяющая начало содержимого пакета. В конец пакета записывается та же последовательность. Эта последовательность не должна встречаться в самом пакете. Т.к. для каждого пакета может быть определена своя последовательность - это ограничение вполне выполнимо. Здесь последовательность для наглядности стоит рассматривать как бинарный аналог обычных скобок. Такой способ объединения реализован например в http-multipart. Из достоинств такого подхода стоит рассмотреть удобство использования при составлении пакетов вручную.
2. Объединение с помощью заголовка фиксированной длины, содержащего размер содержимого пакета. В этом способе после чтения заголовка, в буфер считывается последующие байты в количестве, указанном в заголовке. Такой подход значительно проще в реализации, т.к. требует меньше шагов при парсинге, однако создание пакетов в ручном режиме сильно затрудняется, т.к. требует подсчета размера содержимого.

Формат dataforge-envelope использует второй подход в силу его простоты. Для обеспечения возможности создания пакетов вручную в формате предусмотрено исключение, которое будет описано в последующих подразделах. Заголовком, содержащим размер пакета служит «бинарный заголовок», изображенный на схеме. Т.к. пакет предусматривает содержание текстовых и бинарных данных, в заголовок записываются 2 числа, соответствующие размерам этих частей. Общая длина пакета получается суммированием этих чисел плюс константным размером разделителей, присутствующих в пакете.

После бинарного заголовка в пакете dataforge-envelope идет часть «метаданные», содержащая текстовые метаданные пакета. В качестве формата метаданных потенциально может выступать любой язык разметки, реализующий древовидную структуру. Сам тип языка разметки указывается в бинарном заголовке. На момент написания были реализованы языки XML и JSON, при этом на установке «Троицк ню-масс» в основном используется JSON. Dataforge-envelope не накладывает ограничений на содержание метаданных, это оставляется на усмотрение разработчика системы сбора и анализа.

Замыкает пакет часть «бинарные данные», содержащая бинарные данные. Формат пакета также не накладывает ограничений на их формат в силу большого разнообразия видов данных. Предполагается, что способ парсинга формата бинарных данных будет однозначно определяться метаданными пакета.

Все 3 части пакета dataforge-envelope разделяются специальной последовательностью символов `\r \n`.

Из достоинств формата можно выделить:

- Возможность передачи пакетов низкоуровневыми протоколами связи по типу TCP/IP. Также есть возможность хранения пакетов без изменений средствами файловой системы.
- Возможность создания вложенных структур.
- При использовании совместно с файловой системой формат позволяет использовать каскадные поля метаданных для вложенных подгрупп пакетов.

К недостаткам можно отнести:

- Затрудненность создания пакета вручную. Однако, как уже было сказано ранее, формат предполагает некоторое исключение, облегчающие данный процесс.
- Вольность задания метаданных и формата бинарных данных. Параметры должны быть ограничены внешними правилами. Этот недостаток вызван им разнообразием потенциальных данных и не может быть исправлен принципиально.

Пример сообщения, содержащего информацию о напряжении на спектрометре во время набора, открытый через текстовый редактор nano:

```
#!^@^A@^@Z^UH’^@^A^@^@^@^@^@’^@^@^B^A^@^@’’!#
{
"format_description": "https://...$",
```

```
"programm_revision": "1.1.1-75-gd67324e",
"type": "voltage"
}

2017-11-22T11:15:01 1 15999.9
2017-11-22T11:15:05 1 16000
2017-11-22T11:15:09 1 16000
2017-11-22T11:15:13 1 16000
...
2017-11-22T12:51:41 1 14900
2017-11-22T12:51:45 1 14900
2017-11-22T12:51:49 1 14900
```

Листинг 3.1 — Пример сообщения в формате Dataforge Envelope.

Версии форматов

В формат протокола dataforge-envelope в процессе разработки системы сбора данных вносились исправления, призванные улучшить его и убрать оказавшиеся ненужными элементы. На данный момент существует 3 версии протокола. Т.к. система сбора данных в силу инертности разработки использует все версии, опишем их.

Версии отличаются друг от друга только бинарным заголовком пакета. Поэтому с помощью регулярных выражений или простого сравнения спецсимволов можно однозначно определить к какой версии пакет относится.

0x14000 Для всех версий бинарный заголовок ограничивается открывающей и замыкающей последовательностью спецсимволов длиной 2. Тип сообщения (внутренняя версия) определяется 4 байтами, идущими после открывающей последовательности. В последних версиях это 4 текстовых символа. Однако в первых версиях эти символы были бинарными. Формат 0x14000 - первая версия протокола. Название соответствует 4 бинарным символам типа в hex кодировке.

Бинарный заголовок имеет размер 30 байт. Структура заголовка приведена в таблице 1.

Типы метаданных:

Байты	Значение	Описание
0-2	#!	Открывающая последовательность символов
2-6	unsigned int32	Тип сообщения. Статичное значение 0x14000
6-10	unsigned int32	Время создания сообщения (формат неопределен)
10-14	unsigned int32	Тип метаданных
14-18	unsigned int32	Длина метаданных в байтах
18-22	unsigned int32	Тип бинарных данных
22-26	unsigned int32	Размер бинарных данных в байтах
26-30	!#\r\n	Закрывающая последовательность символов

Таблица 1 — Структура бинарного заголовка версии 0x14000.

- "UNDEFINED_METATYPE": 0x00000000 - неопределенный формат метаданных
- "JSON_METATYPE": 0x00010000 - метаданные в формате JSON
- "QDATASTREAM_METATYPE": 0x00010007 - метаданные в формате QDataStream (встроенный в Qt бинарный парсер)

Типы бинарных данных:

- "UNDEFINED_BINARY": 0x00000000 - неопределенный формат
- "POINT_DIRECT_BINARY": 0x00000100 - массив событий время-амплитуда
- "POINT_QDATASTREAM_BINARY": 0x00000107 - массив событий время-амплитуда, сериализованный QDataStream
- "HV_BINARY": 0x00000200 - массив напряжений вольтметров высоковольтной стойки в бинарном виде
- "HV_TEXT_BINARY": 0x00000201 - ссив напряжений вольтметров высоковольтной стойки в текстовом виде

DF02 При работе с данными оказалось, что формат 0x14000 имеет неиспользуемые в работе поля бинарного заголовка, в частности оказались неиспользуемыми поля прихода пакета и формата бинарных данных - вся необходимая информация содержится в метаданных пакета. Также открывающая последовательность спецсимволов имела коллизии с форматом shebang[39]. К тому же при просмотре файлов через текстовый редактор возникали трудности с прочтением полей бинарного заголовка.

Байты	Значение	Описание
0-2	#~	Открывающая последовательность символов
2-6	unsigned int32	Тип сообщения. DF02 == 0x44463032
6-8	unsigned int16	Тип метаданных
8-12	unsigned int32	Длина метаданных в байтах
12-16	unsigned int32	Длина бинарных данных в байтах
16-20	~#\r\n”	Закрывающая последовательность символов

Таблица 2 — Структура бинарного заголовка версии DF02.

Учитывая эти факторы была разработана версия «DF02» с описанной в таблице 2 структурой полей.

Типы метаданных:

- "UNDEFINED_METATYPE": 0x00000000 - неопределенный формат метаданных
- "JSON_METATYPE": "JS метаданные в формате JSON
- "XML_METATYPE": "XM метаданные в формате XML

DFTL Для отладки системы и учного редактирования пакетов формат dataforge-envelope предусматривает специальную версию только для файлов. В ней в бинарном заголовке не указываются размеры пакетов, а сами части разделяются последовательностями спецсимволов:

- #~DFTL~# - для начала пакета
- #~META~# - для части, содержащей метаданные
- #~DATA~# - для части, содержащей бинарные данные

Флагом завершения пакета является конец файла. Таким образом с помощью версии формата DFTL можно легко создавать корректные пакеты вручную с помощью текстового редактора.

Типы бинарных данных

Из-за разнообразия типов сохраняемых данных и по историческим причинам система сбора данных установки «Троицк ню-масс» использует 3 основных

Байты	Значение	Описание
0-2	unsigned int16	Амплитуда
2-6	unsigned int32	Относительное время (в единицах MADC)
6-7	bool	Флаг корректности

Таблица 3 — Формат записи события в формате MADC.

формата бинарных данных для формирования пакетов. Каждый формат имеет свои достоинства и недостатки и подходит для конкретного типа данных.

Формат данных MADC Данный самодельный бинарный формат, за исключением экономии памяти путем комбинирования чисел в одно поле полностью повторяет формат записи данных старой системы сбора, написанной Задорожным С.В. Структура данных, используемая в формате, в свою очередь повторяет структуру выходных данных, записанных в MADC с электроники детектора. Бинарные данные формата MADC представляют собой последовательно записанные в бинарном виде события, считываемые с детектора. Формат отдельного события описан в таблице 3.

Данный формат не оптимизирован в плане использования памяти, однако он хорошо подходит для своей цели и его использование упрощает процесс миграции старой логики набора при модернизации системы сбора.

Текстовый формат Для упрощения просмотра, для данных медленного контроля, которые обычно имеют небольшой объем, было решено использовать для записи текстовый формат TSV. Т.к. текст может быть прочитан человеком, при анализе не возникнет трудностей с парсингом данных. Пример бинарных данных в текстовом виде (здесь сохранены показания вольтметра в зависимости от времени; формат хранения времени - ISO):

```
2017-11-22T11:15:01 1 15999.9
2017-11-22T11:15:05 1 16000
2017-11-22T11:15:09 1 16000
2017-11-22T11:15:13 1 16000
```

Листинг 3.2 — Пример данных, набираемых с вольтметра.

Комбинированный формат При обновлении электроники детектора и перехода на сбор кадров появилась необходимость в создании единого формата

хранения бинарных данных, позволяющего содержать в одном пакете как события в виде амплитуд, так и в виде кадров. Такая необходимость вызвана особенностями обработки - в наборе могут присутствовать сложные кадры, которые затруднительно достоверно преобразовать в амплитуду. Такие кадры предполагается сохранять в необработанном виде.

В качестве основы для формата бинарных данных был выбран Protocol Buffers 3 версии[40].

Protocol Buffers — протокол сериализации (передачи) структурированных данных, предложенный Google как эффективная бинарная альтернатива текстовому формату XML. Разработчики заявляют, что Protocol Buffers проще, компактнее (от 3 до 10 раз) и быстрее (от 20 до 100 раз), чем XML. Также для Protobuf существуют готовые библиотеки практически для всех популярных языков.

Комбинированный формат, использующийся на установке «Троицк нюмасс» на языке Proto может быть записан в форме:

```
syntax = "proto3";
package Rsh;

message Point {
message Channel {
message Block {
message Event {
uint64 time = 1;
bytes data = 2;
}
message Peaks {
repeated uint64 times = 1;
repeated uint64 amplitudes = 2;
}
uint64 time = 1;
repeated Event events = 2;
Peaks peaks = 3;
}
uint64 num = 1;
repeated Block blocks = 2;
}
repeated Channel channels = 1;
}
```

В данных поканально записаны выделенные из блоков события. События могут быть храниться в обработанном и необработанном видах. Для обработанных событий записываются время и амплитуда, для необработанных - время и кадр.

Описанная структура дает порядка 24% оверхеда для обработанных событий и 3% для необработанных.

Прозрачная архивация

Формат предусматривает функцию автоматической архивации бинарной части данных при сериализации пакета. Метаданные при таком сжатии остаются доступны для чтения и могут быть использованы для работы с пакетом.

Соглашение о маркировке сжатых бинарных данных:

- В корневой части метаданных должно присутствовать поле "compression"со значением "zlib"
- Бинарные данные должны быть сжаты с использованием библиотеки zlib

Данные условия автоматически выполняются при формировании пакета с помощью реализации протокола dataforge-envelope на Python. Реализация использует библиотеку zlib[41]. В ридми репозитория python-df-parser[42] есть пример создания пакета с сжатой бинарной частью.

Структура хранения данных на HDD

Dataforge-envelope в связке с хранением посредством файловой системы предполагает хранение файлов с выполнением следующих правил:

1. Минимальной единицей информации является пакет формата dataforge-envelope, записанный в отдельный файл.
2. Группы файлов могут быть скомбинированы в одной папке. Для папки предусмотрен специальный файл с названием metadata также имеющий формат dataforge-envelope без бинарных данных и содержащий общие для группы параметры.

3. Группы (папка с файлами пакетов) также могут быть сгруппированы по средствам объединения в одну папку.
4. Поле в файле метаданных группы распространяются на все вложенные файлы, если в них нет соответствующего переписанного поля.

Для датасетов, набираемых на установке «Троицк ню-масс» структура сохраняемых данных реализована по правилам:

1. Все данные хранятся в одной корневой папке.
2. Корневая папка содержит подгруппы с проводимыми экспериментальными сеансами. Названия сеансов совпадают с датой (год и месяц) их проведения.
3. Группа сеанса содержит подгруппы наборов, разделенных логически по параметрам (тестовые, разные стадии напуска трития в установку). Подгруппы называются филами. Название фила устанавливается пользователем.
4. Группа фила содержит подгруппы, соответствующие набору данных при одинаковом сценарии набора. Эти подгруппы называются сетами. Название сета генерируется автоматически по его номеру.
5. Сет содержит:
    а) Файлы набора событий при заданном напряжении спектрометра. В названии файла присутствуют основные параметры - время набора и желаемое напряжение на спектрометре.
    б) Файл метаданных сета, содержащий описание, комментарии и возникающие при наборе ошибки.

Реализации протокола DataForge-envelope

Созданные на момент написания реализации протокола указаны в таблице 4.

Язык	0x14000	DF02	DFTL	zlib
C++ (Qt)	+			
Python	+	+		+
Java	+	+	+	+

Таблица 4 — Реализации формата Dataforge Envelope.

3.2 Модернизированная система сбора

В процессе разработки новой системы сбора данных для установки «Троицк ню-масс» были сформулированы паттерны разработки, направленные на увеличение продолжительности срока службы работы программного комплекса и упрощение поддержки и дальнейшей модернизации:

- Использование микросервисной архитектуры. Каждый самостоятельный, логически полный модуль системы должен быть реализован в виде отдельной программы (сервиса), которая выполняет только свою задачу и взаимодействует с другими модулями системы по TCP/IP протоколу используя стандартизированный формат сообщений DataForge envelope. Такой подход инкапсулирует всю работу, связанную со взаимодействием с аппаратной частью в отдельные блоки, которые возможно разрабатывать параллельно и независимо друг от друга. Стандартизированный интерфейс взаимодействия обеспечивает легкость модернизации или замены модулей.
- Все сервисы должны иметь режим виртуального устройства. В этом режиме сервис должен запускаться на любом ПК и симулировать реальные ответы при запросе. Выполнение этого требования обеспечивает возможность тестирования отдельных элементов без полного включения системы. Кроме того появляется возможность удаленной разработки комплекса. В случае установки «Троицк ню-масс», где в качестве ПК для сервисов используются довольно слабые CCPC7[8], возможность разработки не на используемой в эксперименте аппаратной части сильно увеличивает скорость разработки.
- Каждый сервис должен использовать только необходимую для работы информацию и не требовать дополнительных данных. Это должно максимально изолировать сервисы друг от друга и обеспечить независи-

мость их работы. В процессе эксперимента могут появляться проблемы, как например неполадки в сети. При таких проблемах монолитные комплексы не смогут работать. Описанное требование призвано обеспечить возможность работы комплекса в условиях сбоев.

– Взаимодействие между сервисами должно обеспечиваться одной главной программой. Эта же программа должна предоставлять пользовательский интерфейс и возможность управления набором. Также, для обеспечения стабильности работы в случае поломки компьютера, программа должна быть мультиплатформенной и запускаться на всех ПК. Сформулированное требование вытекает из предыдущего пункта. Описанная программа инкапсулирует в себе все взаимодействие между сервисами т.о. обеспечивая независимость их работы.
– В отладочных целях, все сервисы и подпрограммы комплекса должны вести логи работы и сохранять их на жесткий диск. Все ошибки и предупреждения при наборе, в случае их появления, должны быть выведены пользователю.

С учетом изложенных требований для новой системы сбора данных была выбрана архитектура, соответствующая схеме на рисунке 3.2, где:

– API сервер модуля высокого напряжения - сервис, находящийся на CCPC7 стойки высокого напряжения. Отвечает за установку и коррекцию запирающего напряжения спектрометра.
– API сервер модуля детектора - сервис, находящийся на CCPC7 детекторной части установки. Отвечает за включение набора приходящих на детектор событий.
– GUI клиента - управляющая программа. Отвечает за запуск и управление сценариями набора данных, мониторингом системы набора и сохранением данных.
– Bash скрипты - набор скриптов, отвечающих за синхронизацию локальной БД с БД сервера данных.

В качестве фреймворка разработки комплекса выбран Qt. Для устанавливаемых на CCPC7 серверов используется Qt 4.8. Для клиентской части используется версия Qt 5.4. Также используются дополнительные сторонние библиотеки QJson, QCustomPlot и easylogging++. Код хранится в репозитории bibucket единым проектом.

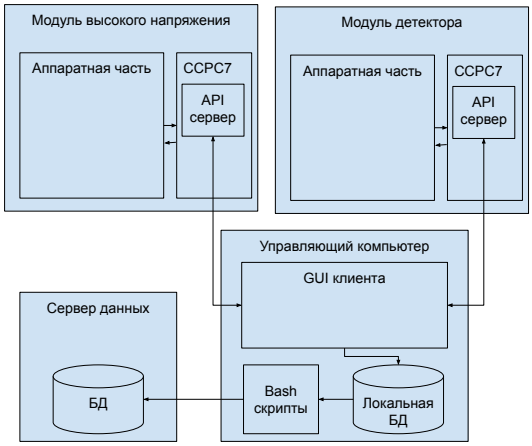

Рисунок 3.2 — Архитектура распределенной системы сбора данных.

3.2.1 Общее для подмодулей комплекса

Логирование

Все модули используют в качестве логгера библиотеку easylogging++. Ее пресет находится в корне проекта в файле easylogging.pri. В конфигурационных файлах сервисов (.pro) логгер подключается командой:

```
include(../common.pri )
```

Листинг 3.3 — Подключение пресета common.

Также в main.cpp сервиса для корректной работы прописаны команды:

```
#include <easylogging++.h>
#include <common.h>

#ifdef EL_CPP11
INITIALIZE_EASYLOGGINGPP
#else
_INITIALIZE_EASYLOGGINGPP
#endif

int main(int argc, char *argv[])
{
setCodecs();
initLogging(argc, argv);
```

```
logModes();
```

Листинг 3.4 — Инициализация пресета common.

В пресетах common.pri содержится код для:

– Инициализации логгера.
– Инициализации временной папки. Создаваемая временная папка в процессе работы сервиса мониторит свой размер и, при превышении порога, удаляет файлы в порядке их создания.

TCP

В пресете tcp.pri, расположенном в папке tcp находится код для парсинга dataforge-envelope протокола а также коды TCP клиента и API сервера, работающего на этом протоколе.

Обработка сообщений клиентом и сервером осуществляется с привязки сигналов

– receiveMessage - получение сообщение. В привязанном к этому сигналу слоте выполняется код по работе сервиса в соответствии с содержимым входного пакета.
– sendMessage или sendRawMessage - отправка ответа
– error - отправка ошибки

Такая абстракция позволяет инкапсулировать код взаимодействия с транспортными протоколами в классе-предке и сосредоточиться только на обработке содержимого пакетов.

Последовательность работы и подключения

Все сервисы программного комплекса рассчитаны на длительное единоразовое подключение к сокету. После обработки команды сервисы не будут закрывать соединение с клиентом.

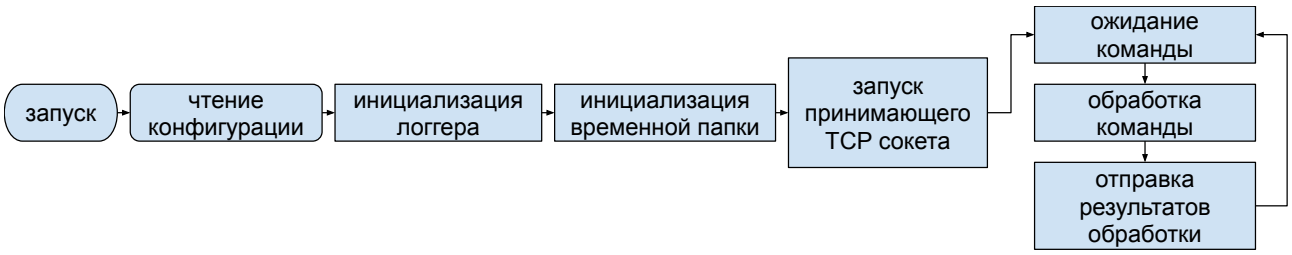

Рисунок 3.3 — Схема работы сервиса детектора.

Т.к. логика работы набора данных подразумевает последовательное исполнение, в целях упрощения системы все сервисы рассчитаны только на строго последовательное выполнение команд. В случае, если в процессе обработки сервисом очередной команды поступит пакет с новой командой - в ответ на него сервер передаст ошибку :

```
{  "type": "reply",
"reply_type": "error",
"error_code": "8",
"block" : "1",
"stage": "check busy",
"description": "1 busy" }
```

Листинг 3.5 — Пример сообщения об ошибке (сервис занят).

Здесь error_code - код ошибки, stage - стадия, на которой ошибка произошла, description - описание ошибки.

3.2.2 Модуль детектора

Общая схема работы сервиса представлена на рисунке 3.3.

В целях достижения гибкости настройки адреса всех модулей КАМАК могут быть заданы через конфигурационный файл, который считывается на стадии «чтение конфигурации». В случае некорректной конфигурации приложение выведет соответствующие предупреждения в консоль. Также, в целях дублирования во избежание потери данных при инициализации сервис создает временную папку, в которую будут записываться последние набранные файлы.

Взаимодействие с КАМАК

Сервис взаимодействует с модулями крейта по средствам API предустановленной в ОС CCPC7 библиотеки взаимодействия с КАМАК.

Основной задачей сервиса детекторного модуля является запуск записи событий на блоке КАМАК MADC и их считывание после набора. Полная последовательность команд КАМАК для набора событий за заданное время выглядит можно описать последовательностью:

1. TTL_NIM, команда 0, 17, 0xFFFF - отключение TTL на MADC
2. MADC, команда 0, 12, none - отключение набора на MADC
3. MADC 0, 17, time - запись времени набора в MADC
4. COUNTER1, команда 0..6, none - обнуление значений с 0 по 6 на первом каунтере
5. COUNTER2, команда 0..6, none - обнуление значений с 0 по 6 на втором каунтере
6. MADC, команда 0, 11, none - включение набора на MADC
7. OV1, команда 0, 16, 0x1F - включение всех выходов на OV1, верхний канал
8. OV1, команда 0, 16, 0x0F - включение всех выходов на OV1, нижний канал
9. OV1, команда 0, 25, none - запуск OV1 в импульсном режиме. После этой команды начинается аппаратный набор событий
10. Цикл опроса MADC - завершается при выставлении флагов завершения или переполнения
    а) Ожидание 1 с
    б) MADC, команда 1, 1 - считывание количества набранных событий, флагов завершение набора или переполнения буфера
11. MADC, команда 0, 12, none - отключение набора на MADC
12. MADC, команда 1, 1 - считывание полного количества набранных событий
13. Цикл чтения событий
    а) MADC, 0, 0, none - считывание младших 2 байт времени события

б) MADC, 1, 0, none - считывание старших 2 байт времени события

в) MADC, 2, 0, none - считывание амплитуды и флага корректности события события

Также предоставляет метод API по инициализации крейта КАМАК, которая последовательно выполняет команды Z и C.

Инициализация КАМАК

Перед работой с сервисом необходимо проинициализировать крейт детектора. Для этого необходимо подключиться к серверному сокету сервиса и передать dataforge-envelope пакет с пустой бинарной частью и с следующими полями в корне метаданных:

```
{ "command_type" : "init",
"type" : "command" }
```

Листинг 3.6 — Команда иниализации крейта КАМАК детектора.

Здесь type - тип пакета. Для всех команд значение этого поля должно быть «command»; command_type - тип команды. В данном случае это код команды инициализации - «init».

При получении сообщения сервис последовательно выполнит команды крейта Z и C. После успешного выполнения сервис передаст ответ:

```
{ "type" : "reply",
"reply_type" : "init",
"reseted" : "1",
"status" : "ok" }
```

Листинг 3.7 — Сообщение об успешной иниализации сервиса детектора.

Здесь type - тип ответа. В системе сбора данных «Троицк ню-масс» для ответов это поле может иметь только значение reply, reply_type - тип ответа, status - статус команды (практически для всех команд значение равно ok),

reseted - флаг перезагрузки сервера (0 - если сервер проинициализирован в первый раз, 1 - в противном случае).

3.2.3 Набор событий с детектора

После инициализации крейта детектора, с помощью сервиса можно начать проводить набор событий, регистрируемых детектором. Для этого на сервис необходимо передать пакет с метаданными:

```
{  "acquisition_time" : "5",
"command_type" : "acquire_point",
"type" : "command",
"split" : "true",
"external_meta": {
"HV1_value": "-1",
"HV2_value": "-1",
"acquisition_time": "5",
"point_index": "0" } }
```

Листинг 3.8 — Пример команды на набор событий.

Здесь:

- acquisition_time - Время набора в секундах.
- split - флаг разделения набора на поднаборы по 5 с.
- external_meta - Опциональный контейнер для внешних метаданных. Предназначен для сохранения информации о точке, полученной извне (например напряжение на спектрометре при наборе). Поле будет полностью скопировано в такой же тег в ответе.

При обработке такой команды сервис выполнит, описанную в параграфе 3.2.2 последовательность действий для набора событий в течение фиксированного времени. В случае, если в команде указывается флаг разделения набора, описанная последовательность будет выполнена несколько раз для времени 5 с и общим временным покрытием равным заданному времени набора. Производить набор с разделением следует в случае высокой скорости счета и больших желаемых времен набора, т.к. разделение позволяет избежать переполнение памяти MADC при наборе.

В случае набора без разделения сервис выдаст ответ по типу:

```
{   "binary_size": "2345",
"end_time": "15:56:27.109",
"external_meta": { "HV1_value": "-1",
"HV2_value": "-1",
"acquisition_time": "5",
"point_index": "0" },
"format_description": "https://...",
"programm_revision": "1.59aa102",
"reply_type": "aquired_point",
"start_time": "15:56:22.065",
"status": "ok",
"time_coeff": 50,
"total_events": "335",
"type": "reply" }
```

Листинг 3.9 — Сообщение с набранными событиями (набор без разделения).

Здесь: binary_size - размер полученных бинарных данных в байтах, start_time - приблизительное время начала набора точки, end_time - Приблизительное время окончания набора точки, time_coeff - коэффициент преобразования из внутреннего времени в наносекунды, total_events - количество сосчитанных событий в точке, external_meta - контейнер для внешних метаданных, format_description - ссылка на описание формата бинарных данных, programm_revision - версия программы, с помощью которой набрана точка.

Бинарные данные в ответном пакете будут содержать параметры событий в бинарном формате MADC. Для набора с разделением ответное сообщение будет немного отличаться:

```
{  "acquisition_time": 10,
"binary_size": "455300",
"end_time": [
"2017-11-22T11:15:09",
"2017-11-22T11:15:15"],
"events": [ 32359, 32483 ],
"external_meta": { "HV1_value": "-1",
"HV2_value": "-1",
"acquisition_time": "10",
"point_index": "0" },
"format_description": "https://...",
"programm_revision": "1.1.1-70-gd158942",
"reply_type": "aquired_point",
"split": true,
```

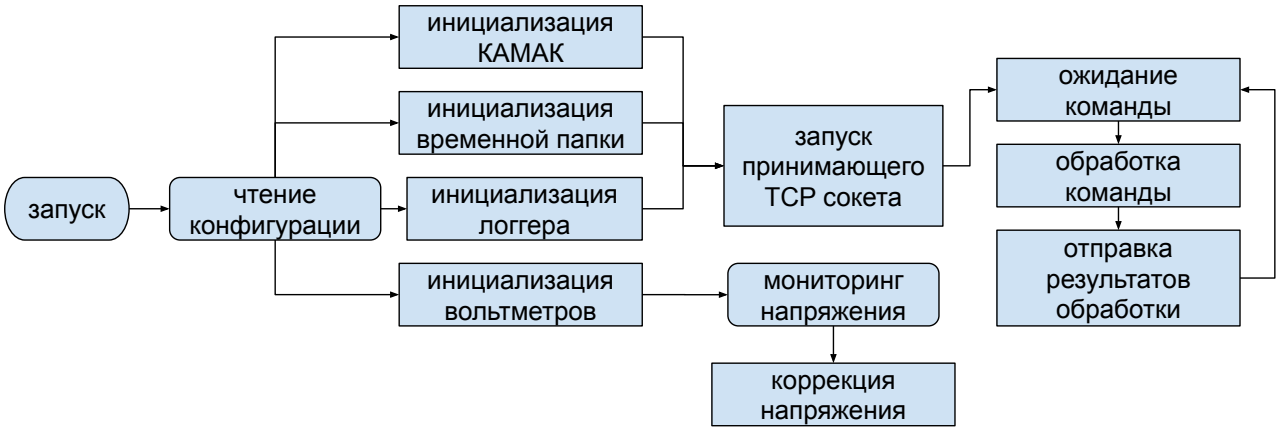

Рисунок 3.4 — Схема работы сервиса высоковольтной стойки.

```
"start_time": [
"2017-11-22T11:15:04",
"2017-11-22T11:15:09" ],
"status": "ok",
"time_coeff": 50,
"total_events": "64842",
"type": "reply" }
```

Листинг 3.10 — Сообщение с набранными событиями (набор с разделением).

Различия: start_time и end_time теперь имеют векторное значение соответствующее временам набора 5 секундных блоков, добавлено поле events соответствующее количеству событий в каждом блоке. Бинарная часть имеет такой же формат, что и в ответе на набор без разделения. По количеству событий в каждом из блоков можно восстановить отступы в бинарных данных.

3.2.4 Модуль стойки высокого напряжения

На рисунке 3.4 изображена схема работы сервиса. Аппаратно, сервис соединяется с блоками питания с помощью:

1. ЦАП, управляемого через КАМАК и порта RS-232 самого БП для основного блока питания. Необходимость использования КАМАК для взаимодействия с ЦАП вызвана отсутствием другой аппаратной части, обеспечивающей задание значений необходимой битности. По сути КАМАК в этом сервисе используется только для программного управления ЦАП.

2. Порта RS-232 для блока питания смещения.

Порты RS-232 подключаются к CCPC7 через по USB через конвертеры. Аналогично к CCPC7 через RS-232 подключаются вольтметры.

Сервис может работать в 2-х режимах выставления напряжения:

1. Выставление напряжения через 2 блока питания. В таком режиме сначала выставляется базовое напряжение и затем с помощью первого блока питания через ЦАП и com порт БП. Затем от базового напряжения вычитается напряжение на втором блоке питания. Т.к. второй блок питания может изменять напряжения в пределах 1 кВ, такой режим подходит для проведения прецизионных измерений в небольшом диапазоне энергий.
2. Выставление напряжения с помощью одного блока питания. Этот режим аналогичен предыдущему за исключением отсутствия второго шага и подходит для набора данных в широком диапазоне энергий.

Необходимый режим работы сервиса можно задать через конфигурационный файл.

При запуске приложения, после чтения конфигурации сервис также как и в модуле детектора проводит инициализацию логгера и временной папки. Далее автоматически проводится инициализация КАМАК, при которой на вход ЦАП выставляется значение 0. Инициализация здесь происходит автоматически для минимизации рисков нежелательного выставления напряжения на блок питания, т.к. начальные значения при старте КАМАК отличны от нуля. Параллельно происходит инициализация вольтметров. В режиме работы с одним блоком питания инициализируется только один вольтметр - второй может быть использован вручную для сторонних измерений; в режиме с двумя БП- инициализируются оба вольтметра. После успешной инициализации вольтметров запускается отдельный программный поток, осуществляющий мониторинг и коррекцию выставленного напряжения.

Коррекция напряжения

В процессе эксплуатации системы выяснилось, что выставляемое через вольтметры напряжение в действительности задается с плавающей ошибкой

порядка 7 В. Для устранения этой ошибки в работу сервиса был встроен алгоритм коррекции. Описать его можно следующей последовательностью:

1. При установке напряжения через ЦАП устанавливается основная часть напряжения минус 100 В. Оставшиеся 100 В устанавливаются через RS-232 и обеспечивают запас для коррекции.
2. При измерении напряжений на вольтметре сервис записывает во внутренний буфер последние 3 измерения.
3. При каждом новом измерении напряжения на вольтметре, если разница между текущим и предыдущим измерениями оказывается меньше заданного порога (т.е. напряжение стабилизировано) и при этом измеренное напряжение отличается от желаемого – к 100 В, устанавливаемым через RS-232 добавляется или вычитается значение, равное разнице между измеренным и желаемым напряжениями.

Описанный метод коррекции напряжения позволяет добиться стабильной точности при выставлении порядка десятых долей Вольта.

Инициализация вольтметров

Инициализация и управление вольтметрами происходит через RS-232 порт и описывается последовательностью действий:

1. Перевод вольтметра в режим программного контроля.
2. Выставление необходимой точности.
3. Выставление мода медленного изменения напряжения.
4. Выставление сопротивления 10 ГОм.
5. Выставление конфигурации для медленных измерений.

После инициализации сервис в цикле проводит последовательное чтение напряжения. Считанные данные отправляются клиенту, если он подключен, автоматически без запроса на измерение.

Выставление напряжения

После подключения, через сервис можно задать напряжение на блоке питания. Для этого по сокету необходимо отправить пакет, содержащий метаданные:

```
{  "block": "1",
"command_type" : "set_voltage",
"type" : "command",
"voltage": "16.1818" }
```

Листинг 3.11 — Команда на выставление напряжения спектрометра.

Здесь block - номер блока питания (1 - основной, 2 - блок смещения), voltage - желаемое напряжение в Вольтах.

При получении команды, сервис выставит через ЦАП и RS-232 необходимое напряжение и начнет проводить непрерывную коррекцию к заданному уровню. После достижения требуемого напряжения по каналу сокета будет передан ответ:

```
{  "type": "answer",
"block" : "1",
"answer_type" : "set_voltage",
"status": "ok" }
```

Листинг 3.12 — Сообщение об успешном выставлении напряжения на спектрометра.

Сервис также имеет возможность задания напряжения с ожиданием его выставления. Запрос такой команды немного отличается:

```
{  "block": "1",
"command_type" : "set_voltage_and_check",
"type" : "command",
"voltage": "16.1818",
"max_error": 5,
"timeout" : 20 }
```

Листинг 3.13 — Команда на выставление напряжения спектрометра с последующей проверкой.

Здесь max_error - допустимая ошибка в Вольтах, timeout - максимальное время ожидания в секундах.

Ответ на команду придет только после того как измерения вольтметра будут совпадать с желаемым значениям в пределах погрешности.

3.2.5 Управляющий модуль

На клиентском компьютере, с которого осуществляется управление набором данных должен быть установлен управляющий программный комплекс системы сбора данных. Он состоит из GUI управления и GUI визуализации данных. Т.к. используемая при разработке библиотека QCustomPlot имеет утечку памяти, программу управления пришлось разделить на 2 подпрограммы во избежания вылетов во время набора.

Управляющая программа позволяет:

1. Подключаться к сервису детектора и проводить набор событий за фиксированное время вручную.
2. Подключаться к сервису высоковольтной стойки задавать и измерять напряжение на блоках питания вручную.
3. Проводить серии наборов событий, регистрируемых детектором по сценарию. Шагами сценария могут быть:
    а) Выставление напряжения на блоке питания спектрометра,
    б) Выставление напряжения на блоке питания спектрометра с последующей проверкой показаний вольтметра,
    в) Запись событий, регистрируемых детектором за фиксированное количество времени,
    г) Второй и третий подпункты одновременно (набор точки).
4. Для сценариев, содержащих только команды по набору точек возможен набор обратным проходом.
5. Сценарий может быть выполнен несколько раз, в т.ч. с чередованием проходов.

Глава 4. Расширения системы сбора

За время эксплуатации новой системы сбора данных периодически появлялась необходимость в изменении ее конфигурации, добавлению и замене элементов. Благодаря выбранным в начале разработки паттернам проектирования, все необходимые модификации проводились относительно просто. В этой главе будут описаны два основных расширения системы, на примере которых можно посмотреть архитектуру на в действии и оценить ее эффективность.

4.1 Интеграция Лан10-12PCI

Переход установки «Троицк ню-масс» на поиск стерильных нейтрино требует повышения скорости счета в системе регистрации событий. Для выполнения этого требования было разработано расширение системы, добавляющее новый модуль обработки сигнала детектора, основанный на АЦП Лан10-12PCI.

В целях тестирования, оба модуля были подключены к сигналу детектора и настроены на синхронный набор данных. На время отладки нового модуля Лан10-12PCI, дублирующие данные MADC использовались для контроля качества и оценки корректности обработки. Схема подключения представлена на рисунке 4.1.

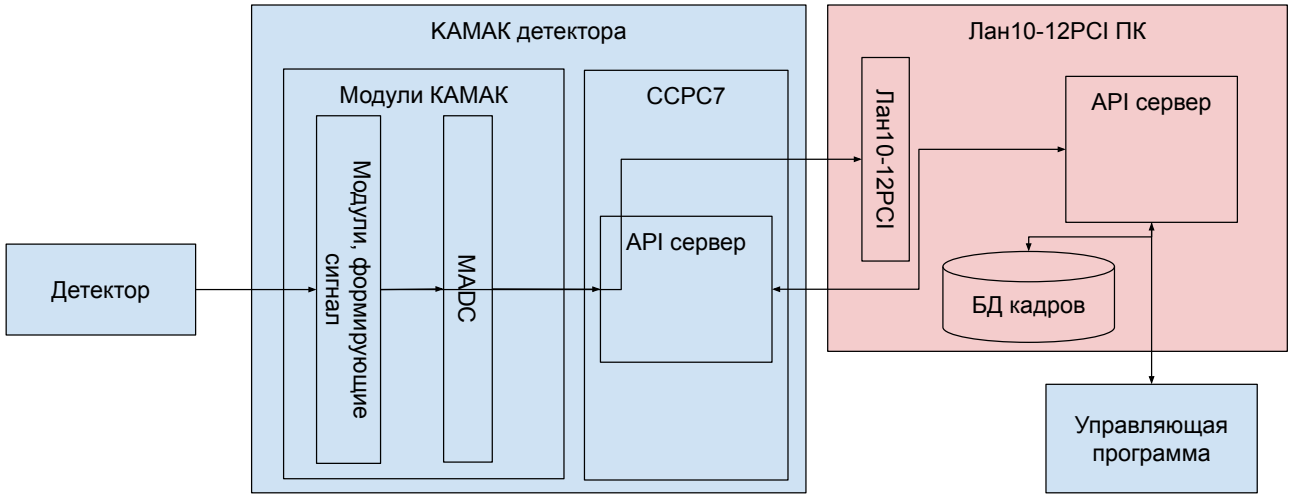

Рисунок 4.1 — Подключение платы Лан10-12PCI. Розовым цветом обозначены новые модули системы.

Поступающий с детектора сигнал после формирования модулями КАМАК детектора делится и теперь кроме входа в MADC также входит в плату АЦП Лан10-12PCI, которая установлена в отдельном компьютере. На этом же компьютере запущен API сервис протокол сообщений которого полностью совпадает с протоколом API сервера, установленного на CCPC7. Для набора с обоих модулей управляющей нужно переподключиться к новому API серверу, если же не делать этого - то при наборе будет работать только старая система.

Новый API сервер работает как «сниффер» между стандартным детекторным блоком и управляющей программой. При получении команды от управляющей программы, сервер передает ее в CCPC7 без изменений. Ответы на команды, приходящие от КАМАК также прозрачно транслируются обратно в управляющую программу. В случае, если через сервер проходит команда на набор событий - помимо передачи команды на CCPC7 программа также запустит набор кадров через плату Лан10-12PCI с таким же временем набора. После завершения набора обеими системами на управляющую программу программу будет передан ответное сообщение аналогичное ответу на набор точки API сервера CCPC7. Т.к. набираемые новой системой кадры получаются значительно больше данных MADC в целях экономии передаваемого трафика и минимизации проблем, создаваемых высокими нагрузками на сеть, данные Лан10-12PCI после набора сохраняются локально на жесткий диск ПК. Синхронизация с общей БД проводится один раз после завершения сеанса сбора данных.

Архитектура системы сбора разделяет работу по самостоятельным компонентам и инкапсулирует всю специфичную обработку внутри, выводя во вне только программно-независимый API. Благодаря этому описанное расширение системы потребовало создать всего один новый сервис. Существующие компоненты при этом остались без каких-либо изменений. Для работы в дублирующем режиме в управляющей программе необходимо только изменить ip адрес и порт сервиса детектора. Также просто можно отключить новую систему, что может быть полезно в случае появления неполадок в ее работе.

4.1.1 Python-df-tcp

Для ускорения разработки сервиса, было решено реализовать программу на языке Python. Исходники можно найти в репозитории [43]. Сам сервер создан на базе подпроекта python-df-tcp[44], реализующего виртуальные классы сервисов использующих в качестве протокола обмена сообщениями формат dataforge-envelope. Проект python-df-tcp использует библиотеку Python 3, asyncio и предоставляет базовые классы для сервера и клиента. В теле базового класса реализован функционал по приему, склейке и отправки сообщений обратно по TCP/IP подключению.

Пример реализации эхо-сервера на базе df-tcp будет выглядеть:

```
import asyncio

from dftcp import DataforgeEnvelopeProtocol


class EchoServerProtocol(DataforgeEnvelopeProtocol):
"""Echo server protocol."""

def process_message(self, message):
"""Send original message back to receiver."""
print("receive {}, {}".format(
message['meta'], message["data"]))
self.send_message(
message['meta'], message["data"], 0)


if __name__ == "__main__":
loop = asyncio.get_event_loop()
coro = loop.create_server(EchoServerProtocol, "0.0.0.0", 5555)
server = loop.run_until_complete(coro)

print('Serving on {}'.format(server.sockets[0].getsockname()))

try:
loop.run_forever()
except KeyboardInterrupt:
print("Programm stoped by user input")
```

```
server.close()
loop.run_until_complete(server.wait_closed())
```

Как видно по приведенному скрипту весь процесс обработки сообщений упакован в переопределенный метод process_message, который, с помощью метода send_message базового класса DataforgeEnvelopeProtocol отправляет приходящее сообщение обратно отправителю. В условии __main__ приведен код для стандартно запуска сервера на asyncio.

Пример реализации скрипта эхо-клиента на базе df-tcp выглядит:

```
import asyncio

from dftcp import DataforgeEnvelopeEchoClient as DFClient


def callback(message, client_obj):
"""Print received message."""
print("receive {}, {}".format(message["meta"], message["data"]))
client_obj.transport.close()


if __name__ == "__main__":
loop = asyncio.get_event_loop()
coro = loop.create_connection(lambda: DFClient(
loop=loop, meta=dict(field="value"), data=b"binary",
callback=callback, timeout_sec=30), "localhost", 5555)
loop.run_until_complete(coro)
loop.run_forever()
```

Здесь задание передаваемых данных и параметров подключения устанавливается методом create_connection в условии __main__. Функция callback определяет обработку ответа от сервера. В примере это вывод содержимого пакета в консоль и последующее закрывание канала сокета.

Описанный примером функционал позволяет создавать на базе модуля df-tcp сервисы, использующие протокол dataforge-envelope, в частности сервер осуществляющий прозрачную передачу пакетов на API сервис КАМАК детектора, который будет описан далее.

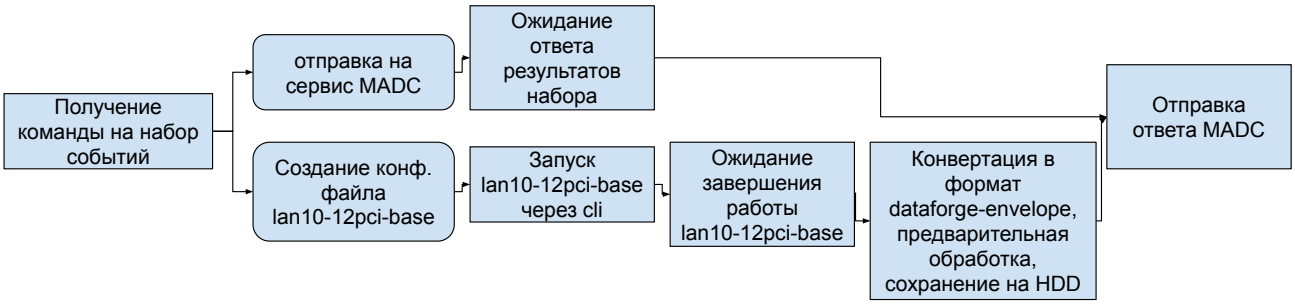

Рисунок 4.2 — Блок-схема работы сервиса Лан10-12PCI.

4.1.2 Набор кадров с помощью Лан10-12PCI

Для работы с платой Лан10-12PCI ранее на C++ было написано консольное приложение lan10-12pci_base, реализующее cli для работы с платой. Следуя DRY принципу, было решено использовать lan10-12pci_base без изменений в качестве средства взаимодействия с платой. Т.к. формат управления приложением отличается от используемого в системе сбора данных «Троицк ню-масс», для работы был написан конвертер, преобразующий выходные данные в формат dataforge-envelope.

Схема работы сервиса Лан10-12PCI в режиме набора событий с детектора изображена на рисунке 4.2.

При получении команды на набор, пакет содержащий ее сразу транслируется в сервис КАМАК детектора. Одновременно с этим во временной директории Лан10-12PCI ПК создается конфигурационный файл имеющий совместимый с lan10-12pci_base формат и содержащий аналогичные с КАМАК параметры набора. После создания конфигурации сервер запускает программу lan10-12pci_base в тихом режиме и передает ей созданный файл. К завершению создаваемого процесса привязан колбэк, который запускает предварительную обработку в виде алгоритма zero-suppression и последующее конвертацию полученных кадров событий в пакет dataforge-envelope и сохранение на диск. После сохранения программа ожидает ответа от сервиса MADC. При получении ответа MADC происходит транслирование его в управляющую программу. Т.к. для создание конфигурационного файла для программы lan10-12pci_base происходит мгновенно, обе системы имеют практически полное пересечение наборов по времени. Конвертация выходных данных программы lan10-12pci_base тоже происходит практически сразу и не вносит дополнительного ожидания по

сравнению с обычным набором. Описанный алгоритм дожидается окончания набора обеих систем, что исключает наложение команд и создание очереди набора, которая может привести к несооветвию результатов работы систем при обработке одной и той же команды.

Файлы, набираемые сервисом Лан10-12PCI отличаются по формату от файлов MADC:

```
{  "bin_offset": 0,
"bin_size": 83483520,
"external_meta": {
// ...
"process_params": { "area_l": 100,
"area_r": 200,
"threshold": 750 },
"params": { /* ...  */ },
"compression": "zlib" } }
```

Листинг 4.1 — Сообщение с набранными платой Лан10-12PCI событиями.

Отличие в отсутвии в корне метаданных полей, относящиеся к событиям, такие как: полное количество зарегистрированных событий, коэффициенты перевода времени. В корне остаются только 2 поля: bin_offset - отсуп в байтах в бинарной части пакета, bin_size - размер бинарной части пакета в байтах. Также в словарь external_meta записываются дополнительные поля: process_params - параметры обработки zero-suppression и params - конфигурация Лан10-12PCI, используемая при наборе и сконвертированная из конфигурационного файла в JSON.

4.1.3 Настройка Лан10-12PCI ПК

Т.к. в течение экспериментального сеанса данные, набранные платой Лан10-12PCI находятся только в одном месте - на Лан10-12PCI ПК - для просмотра и анализа данных в реальном времени была создана программа, реализующая web-интерфейс с возможностью просмотра набираемых файлов. Дизайн интерфейса подобен дизайну DataVisualizer. Бэкенд сервера написан на Python с использованием фреймворка Flask. Бэкенд реализует методы API по

выводу дерева файлов в локальной БД, считыванию метаданных файла набора, построение в реальном времени гистограммы для файла с последующим кэшированием результатов. При генерации гистограммы для преобразования кадров в события используется простой поиск максимума в кадре, что достаточно для визуального представления данных и оптимально по скорости обработки. Фронтенд web интерфейса написан на JavaScript с использованием TypeScript в качестве метаязыка. Для визуализации графических данных используется библиотека Plotly.js. Для остальных элементов интерфейса использовалась библиотека Bootstrap. Интерактивность страницы обеспечивается библиотекой JQuery. Исходный код сервера web интерфейса выложен в репозиторий [45].

На Лан10-12PCI ПК в качестве ОС установлена Ubuntu 16.04. Для удобства использования комплекса сервис Лан10-12PCI и web-интерфейс запускаются автоматически при включении компьютера. Автозапуск реализован с помощью встроенного в Ubuntu сервиса systemd. Для инкапсулирования Python зависимостей в каждом проекте использовался модуль Pipenv. Чтобы обеспечить совместимость генерируемой Pipenv среды и systemd, в конфигурацию сервиса в поле исполняемой команды в качестве интерпретатора Python задается путь к ссылке python, находящейся непосредственно в создаваемой Pipenv папке среды.

В ходе эксплуатации Лан10-12PCI на Linux, выяснилось, что поставляемые компанией Руднев-Шиляев драйвера для Linux ведут себя нестабильно при обновлениях системы - часто при перезапуске Лан10-12PCI ПК и последующем запуске набора оказывалось, что драйвер не видит установленной в компьютер платы АЦП. Скорее всего причиной этой проблемы является несовместимость ранее установленной версии драйвера и новой версии ядра Linux, устанавливаемого при обновлении пакетов. Найденное решение проблемы заключается в полной переустановке всех драйверов RudShel и последующем перезапуске компьютера.

4.1.4 Тестирование набора кадров с помощью платы Лан10-12PCI

Переход на набор непрерывных кадров помимо модернизации аппаратной части системы набора событий требует также проведения тестирования и валидации качества набора новой системы. Для этого на всех сеансах с участием новой платы Лан10-12PCI, набор проводился одновременно двумя системами: с помощью MADC и с помощью непосредственно Лан10-12PCI. Помимо этого, вместе с сигналом детектора к обоим системам был подключен также тестовый генератор, обеспечивающий стабильные периодические импульсы с постоянными частотой равной 50 Гц и амплитудой, величина которой заведомо выше амплитуд набираемых с детектора событий. Эти два условия позволяют проводить сравнение набираемых данных и обеспечивают возможность качественной оценки корректности работы новой системы набора. При сравнении, для выделения параметров событий из набираемых платой Лан10-12PCI кадров был использован второй метод, описанный в главе «Обработка кадров непрерывного сигнала». В описанном далее сравнении будут рассматриваться данные, набранные на сеансе 2017_05.

Поиск функции перехода между MADC и Лан10-12PCI

Для сравнения данных двух систем прежде всего необходимо перевести их в одни единицы измерения. Данные платы Лан10-12PCI имеют особенность - из-за записи реальных 12-битных амплитуд в 16-битные значения, все набранные значения имеют дискретное распределение амплитуд кратное 16. Это накладывает сильные ограничения на возможность преобразований - т.к. при гистограммировании необработанных данных во избежания появления артефактов необходимо выбирать биннинг также кратный 16; если же с данными проводились преобразования, то для подбора правильного биннинга (не создающего артефактов) необходимо сначала определить исходный, кратный 16 биннинг и затем применить к нему то же преобразование что и к данным. В реальной обработке это довольно накладная процедура, т.к. требует дополнительных расчетов. Зная эту особенность и учитывая тот факт, что на данных

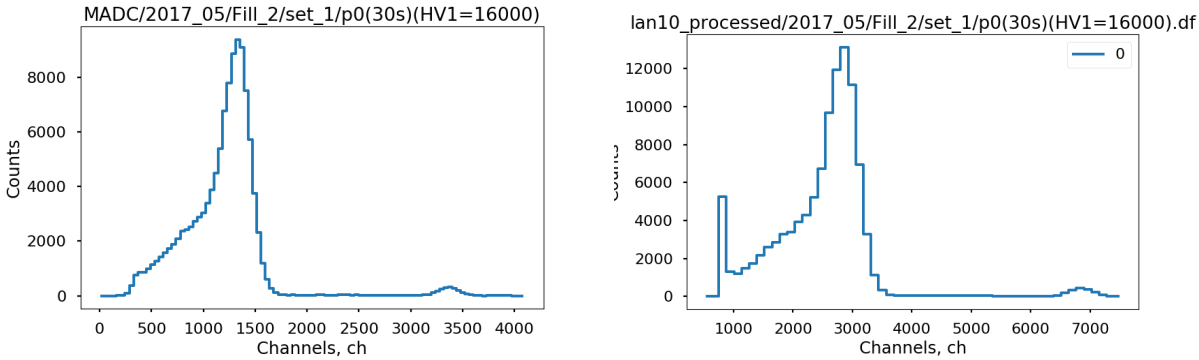

Рисунок 4.3 — Формы спектра одинаковой точки для модуля MADC (слева) и Лан10-12PCI (справа)

MADC такого не происходит, было решено использовать единицы измерения Лан10-12PCI как основные и искать функцию перехода от единиц измерения MADC к Лан10-12PCI, а не наоборот. Т.о. при преобразованиях данные Лан10-12PCI не будут изменяться, что облегчит дальнейший подбор правильного биннинга.

Посмотрим на данные: на графиках 4.3 изображены типичные спектры для мониторных точек набираемые системами MADC и Лан10-12PCI.

Как из них видно набранные спектры для обеих систем визуально совпадают совпадают. Справа от амплитудных спектров реальных событий, поступающих с детектора, можно увидеть небольшой пик - это гистограмма созданных тестовым генератором событий событий. Также можно заметить, что форма амплитудных спектров генератора напоминает функцию Гаусса, а положение максимума которой приходится примерно на 3400 канал для данных MADC и примерно 6800 канал для платы Лан10-12PCI. Предварительно можно сказать, что масштабы шкал будут различаться примерно в два раза. Однако помимо отличий в масштабе единиц изменения данные могут иметь также отличающиеся нули координат. Учитывая вид данных: их визуальное сходство, явную линейность шкал и возможное потенциальное отличие нулей для обеих систем в качестве функции перехода было решено взять полином первой степени со свободным членом. Т.к. в набранных системами файлах самым стабильным и известным объектом является амплитудный спектр событий, создаваемых тестовым генератором, коэффициенты выбранной функции перехода будут рассчитываться именно по нему. Расчет обоих коэффициентов возможен благодаря наличию у спектра аналитической функции обладающей максимумом и шириной.

Алгоритм подбора коэффициентов

Алгоритм поиска коэффициентов между системами можно описать следующей последовательностью операций:

1. Объединение данных сета. Отдельный файл, время набора которого составляет 30 с) из-за малой скорости счета тестового генератора в 50 Гц содержит примерно 1.5к событий, чего недостаточно для точного расчета коэффициентов. Сет в свою очередь содержит 107 точек которые суммарно содержат порядка 160к событий генератора - этого количества уже становится достаточно. Объединение генераторных событий из всех точек сета позволяет построить амплитудный спектр с достаточной точностью.
2. Гистограммирование событий объединенных файлов всего сета.
3. Вырезание из полученных гистограмм для MADC и Лан10-12PCI амплитудных спектров создаваемых генератором событий, т.к. из всего спектра для расчета коэффициентов мы будем пользоваться только ими. При вырезании для обоих спектров в целях сохранения правильности отступов используются абсолютные значения амплитуд относительно нуля.
4. Фитирование обоих пиков функцией Гаусса. Получение значений положений максимума и ширины гауссиана для данных MADC и Лан10-12PCI. Переход от численных спектров к аналитическим функциям с подобранными фитированием коэффициентами обеспечивает большую стабильность алгоритма поиска функции перехода между двумя системами.
5. Вычисление коэффициентов перехода между двумя полученными гауссианами. Полученные коэффициенты будут являться также и коэффициентами перехода между спектрами реальных событий детектора.

Как было сказано ранее, в качестве функции перехода была выбрана линейная функция со свободным членом. Формула перехода данных MADC к Лан10-12PCI может быть выражена формулой $x_m = ax_l + b$, где x_l- амплитуда набранного с помощью Лан10-12PCI события, x_m- амплитуда события MADC, a - коэффициент масштабирования между шкалами MADC и Лан10-12PCI, b - отступ нуля Лан10-12PCI от нуля MADC в единицах измерения MADC.

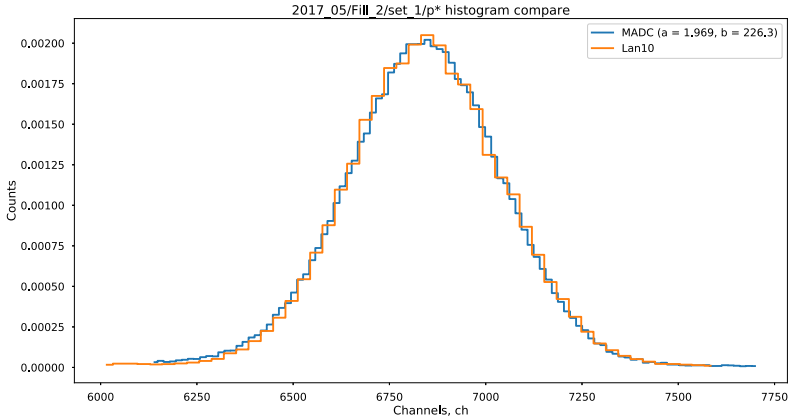

Рисунок 4.4 — Поиск коэффициентов перехода по генераторным пикам.

Для расчета отступа нуля a используется формула $a = \frac{\delta_l}{\delta_m}$, где δ_l, δ_m - среднеквадратичные отклонения построенных фитированием функций Гаусса для Лан10-12PCI и MADC соответственно. Коэффициент масштабирования b вычисляется по формуле $b = \mu_l - \mu_m a$, где μ_l, μ_m - математические ожидания построенных гауссианов Лан10-12PCI и MADC соответственно, a - рассчитанная ранее разница между отступами.

Подбор коэффициентов

При вычислении коэффициентов функции перехода, на этапе гистограммирования для данных MADC использовался размер бина равный 80 каналам и размер бина равный 32 - для платы Лан10-12PCI. На графике 4.4 показан пример амплитудных спектров генераторных событий двух систем, наложенных друг на друга с использованием описанной выше формулы перехода.

Для обеспечения соответствия между разными размерами бинов, на этапе поиска коэффициентов фитированием использовались нормализованные на единицу амплитудные спектры. Как видно из графика форма спектров действительно может быть описана функцией Гаусса, а подобранная функция перехода правильно отображает один спектр на другой. В данном примере единицы измерений различаются в 1.969 раза, а разница между нулями систем составляет примерно 226 каналов MADC.

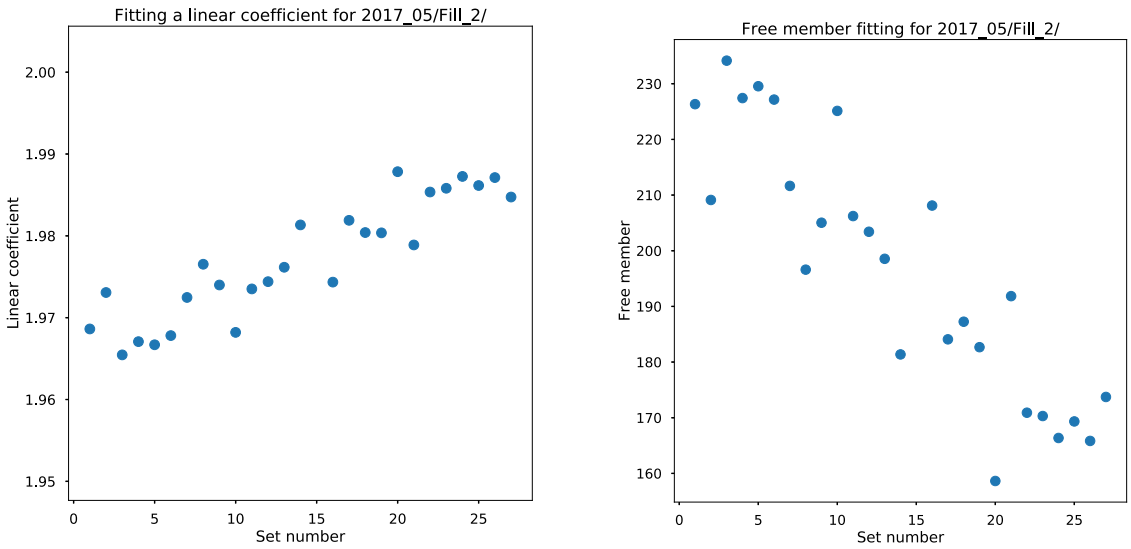

Рисунок 4.5 — Изменение коэффициентов перехода между шкалами MADC и Лан10-12PCI.

Расчет общих коэффициентов перехода между двумя системами предполагалось получить усреднением коэффициентов по всем физическим сетам сеанса, имеющим генераторные события. Однако в процессе нахождения коэффициентов для сетов выяснилось что их значения существенно отличаются друг от друга. С целью выяснения природы различия значений была построена временная зависимость коэффициентов.

Как видно из графиков на рисунке 4.5 значения коэффициентов изменяются не случайным образом: на протяжении набора группы Fill_2 сеанса 2017_05 коэффициент перехода между шкалами MADC и Лан10-12PCI линейно увеличивался со значения 1.96 до примерно 1.99, а разница между положениями нуля линейно сократилась примерно на 60 каналов MADC: с 230 до 170.

Обнаруженное поведение коэффициентов масштабирования в силу неслучайности отклонений не позволяет получить общие коэффициенты для функции перехода между данными MADC и Лан10-12PCI - для обеспечения точного соответствия необходимо рассчитывать коэффициенты по амплитудным спектрам генераторных точек для каждого сета отдельно. Динамика изменений коэффициентов может свидетельствовать о нестабильности электроники: либо системы набора MADC, либо платы Лан10-12PCI. Причина линейного изменения требует дальнейшего анализа и тестирования работы обоих систем. Однако в целом по полученным результатам можно утверждать что новая система сбора данных, основанная на наборе кадров непрерывного сигнала с помощью пла-

ты Лан10-12PCI и дальнейшей обработки программными способами работает стабильно, а получаемые ей спектры с точностью до описанной погрешности совпадают со спектрами старой системы.

4.2 Интеграция DANTE

В рамках сотрудничества TRISTAN, на установке «Троицк ню-масс» был проведен экспериментальный сеанс, на котором было проведено тестирование и отладка прототипа детектора производства лаборатории Макса Планка, Мюнхен[46]. К финальной версии детектора, которая будет использоваться на установке «KATRIN» предъявляются следующие требования:

- Высокая разрешающая способность и возможность задания низкого порога срабатывания (1 кэВ). Требование обусловлено особенностями измерений - свидетельства существования стерильного нейтрино в набираемых данных будут выражаться в слабом отклонении спектра набираемых амплитуд - чтобы достоверно различать такое отклонение требуется разрешающая способность не менее 300 эВ полной ширины на уровне половины амплитуды при наборе энергий 20 кэВ
- Способность обработки высокой скорости счета. Для достоверной проверки гипотезы существования стерильных нейтрино требуется данные имеющие статистическую значимость порядка 10^{-6}. С учетом суммарного планируемого времени набора на установке «KATRIN», составляющего 3 года, для достижения желаемой точности требуется проводить набор событий на скорости счета порядка 10^{-8}. При максимальных допустимых скоростях счета для современных детектор порядка 10^5 событий/с требуется создать сегментированный детектор, состоящий из 1000 ячеек, каждая из которых будет набирать события на скорости 10^5 событий/с.
- Большая площадь детектора. Для избавления от эффектов перераспределения заряда между ячейками диаметр пикселя детектора должен быть не менее 2 мм. В то же время для сохранения высокой разрешающей способности, необходимо добиться низкой емкости пикселя детектора

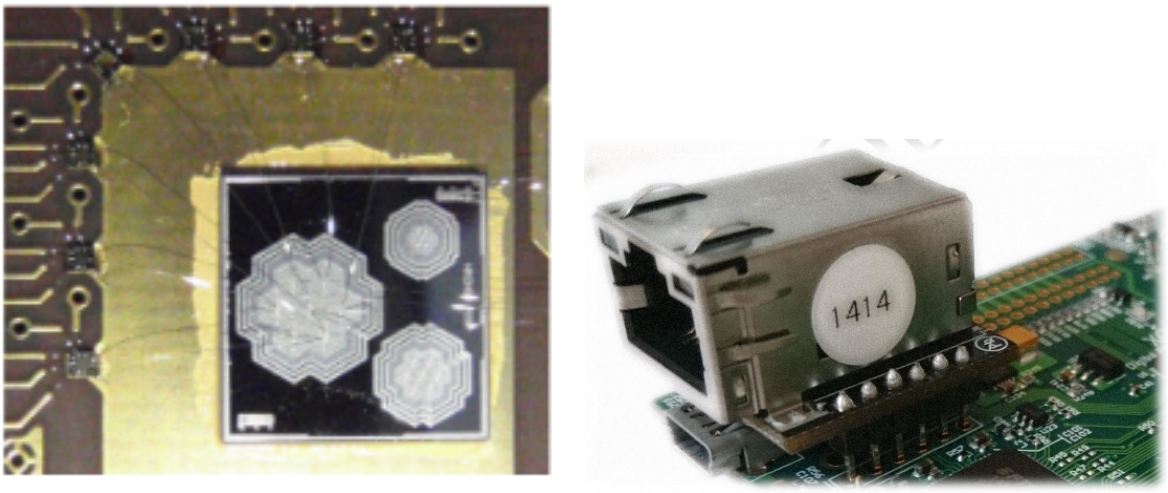

Рисунок 4.6 — Прототип детектора: чувствительная область (слева) и плата DANTE (справа).

С учетом приведенных требований, коллаборацией TRISTAN был создан прототип детектора, представляющий собой дрейфовый кремниевый детектор, сегментированный на 7 гексагональных пикселей. На рисунке 4.6 представлен внешний вид DANTE.

На установке «Троицк ню-масс» проводилось тестирование прототипа с использованием считывающей электроники DANTE разработанной специально под детектор компанией XGLab. В главе будут описаны шаги по интеграции прототипа детектора в систему сбора данных «Троицк ню-масс» и проявившиеся в процессе особенности работы устройства.

Модернизация системы сбора данных установки «Троицк ню-масс», направленная на интеграцию детектора DANTE слегка отличается от модернизации под Лан10-12PCI: т.к. переход на DANTE подразумевает полную замену детектора, из-за этого пропадает необходимость использования всей аппаратуры, отвечающей за набор данных MADC. Вместо прозрачной дублирующей интеграции здесь требуется написать самостоятельный сервис, повторяющий API сервиса детектора системы сбора данных (схема интеграции на рисунке 4.7). Сервис в этом случае является полностью самостоятельным и его возможно установить на отдельный компьютер, чем мы и воспользовались в целях облегчения обратного перехода на использование старого детектора.

Кроме того, т.к. данные будут записываться сразу в последний композитный формат dataforge-envelope и при этом мы не будем иметь продублированного MADC файла, для отладки и анализа набора необходимо также иметь средства визуализации набираемых данных. Для решения последней проблемы

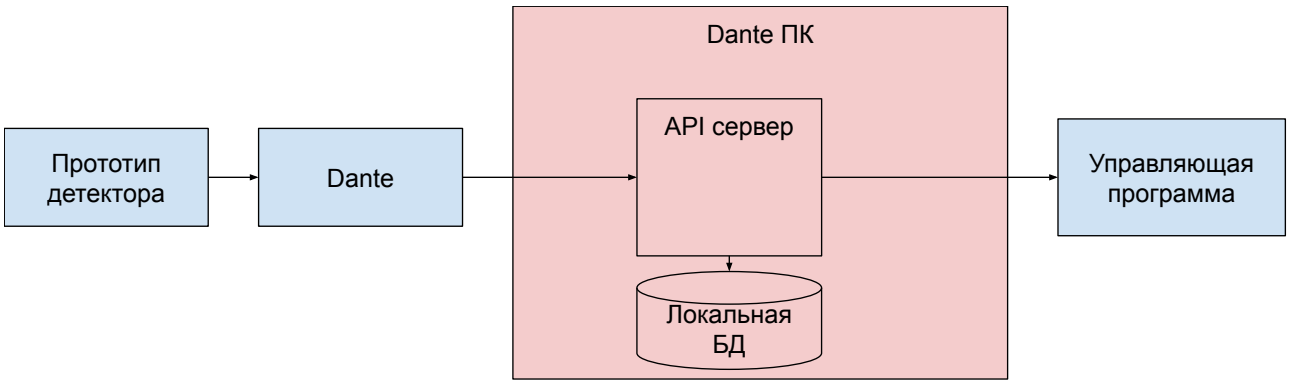

Рисунок 4.7 — Интеграция DANTE в систему сбора данных.

был написан Python скрипт, который устанавливается на управляющий компьютер и посредством alias обеспечивает возможность просмотра с помощью консоли и файловой системы набираемых файлов. Т.к. формат данных такой же как и для Лан10-12PC - скрипт подходит также и для просмотра данных Лан10-12PCI, прошедших обработку кадров.

4.2.1 Взаимодействие со считывающей системой XGLab

Устройство для считывания данных детектора DANTE представляет собой микросхему, главными элементами которой являются зарядочувствительный усилитель подсоединенный к специализированной интегральной микросхеме, осуществляющей оцифровку и обработку (выделение амплитуды и времени события) сигнала. Взаимодействие с DANTE осуществляется посредством TCP/IP сокетов, либо по USB порту с использованием поставляемых с устройством драйверов. Т.к. на момент тестирования существовали только драйвера под Windows, работа с устройством через USB ограничена только этой ОС.

В режиме подключения через TCP/IP DANTE предоставляет 4 сокета с буфером размером 4096 байт и портами (8000-8003), первый из которых является управляющим а остальные - передающими. Инициализация подключения всегда осуществляется клиентом. Все команды устройству должны передаваться через управляющий сокет. Ответы от устройства поступают последовательно и упорядоченно на все порты, начиная с 8000 и заканчивая 8003. По утверждению разработчиков это максимизирует пропускную способность устройства.

Номер	Название	Описание
0	FIRMWARE_VERSION	Версия прошивки Dante
1	DPP_REGISTER_1	Конфигурация оцифровки сигнала
2	DPP_REGISTER_2	Конфигурация оцифровки сигнала
3	DPP_CONFIG_COMMIT_OFFSET	Настройка оцифровки (отступы и задержки)
4	ACQUISITION_STATUS	Параметры набора (старт, стоп и флаг набора)
5	ACQUISITION_TIME	Задание и считывание времени набора
6	ELAPSED_TIME	Считывание пройденного времени набора
7	ACQUISITION_SETTINGS	Выставление параметров набора (режим, маска, биннинг, семплирование)
8	WAVEFORM_LENGTH	Задание и считывание длины кадра
9	MAP_POINTS	Конфигурация сообщений для мониторинга набора
10	TIME_PER_MAP_POINT	Конфигурации частоты прихода мониторных данных
11	ETH_CONFIGURATION_DATA	Задание и считывание параметров подключения по ethernet
12	Reserved	Зарезервированный регистр
13	ETHERNET_COMMIT	Задание и считывание параметров подключения по ethernet
14	CALIB_DONE_SIGNALS	Считывание информации о калибровке плат
15	Reserved	Зарезервировано для последующих модификаций

Таблица 5 — Расположение регистров DANTE.

Электроника DANTE также имеет возможность может быть объединена в цепь, однако на «Троицк ню-масс» тестировалось только одиночное подключение.

Устройство имеет 16 байт конфигурационного регистра. Поля регистров делятся на записываемые, читаемые и записываемые и читаемые одновременно. Управление и контроль DANTE осуществляется через чтение и запись значений в эти регистры. При этом значения полей сырые, т.е. для записи желаемой конфигурации требуется предварительное кодирование параметров в формат, совместимый с соответствующим полем. Т.к. каждый регистр конфигурации содержит в себе несколько параметров, часто, для задания только одного из них необходимо использовать «безопасную запись», т.е. запись в 2 этапа: на первом происходит считывание всего регистра, на втором - изменение конкретного бита и запись всего байта обратно в регистр. Расположение регистров указано в таблице 5

Здесь и во всем бинарном протоколе все биты упакованы в байты с помощью кодировки MSByte.

Протокол взаимодействия

Взаимодействие с DANTE осуществляется с помощью самодельного бинарного протокола. Каждое сообщение имеет бинарный заголовок размером

Байт	Значение	Описание
0	0xAA	Спецсимвол начала пакета
1	0xEE	Спецсимвол начала пакета
2	0xD + CMD(4 bits)	Код команды
3	Board	Номер платы
4	Packet number	Номер пакета. Используется для трекинга ответов
5	Start address	Адрес регистра DANTE к которому применяется команда
6	Length	Количество регистров, к которым применяется команда
7	0x0	Зарезервировано для последующих модификаций

Таблица 6 — Формат заголовка команды DANTE.

8 байт и опциональную часть, содержащую бинарные данные. Сообщения разделяются на команды (посылаемые клиентом устройству) и пакеты (передаваемые устройством на клиент). Оба типа сообщений имеют слегка различные форматы заголовков. Формат команды описан в таблице 6

Сообщение с командой реализует функционал чтения (код команды 0x0) и записи (код команды 0x1) данных из/в регистры конфигурации. 3-й байт в заголовке задает номер платы, к которой будет применена команда. 5-й и 6-й байты задают адрес регистра и количество идущих после адреса регистров, к которым будет применена команда. Оба поля могут содержать значения в диапазоне 0..15. В случае записи, после заголовка должна идти бинарная часть имеющая размер, равный 6-му байту заголовка и содержащая данные для записи в регистры.

Т.к. передача данных по сокету происходит только после записи 2920 байт в буфер, для передачи команды в устройство после записи пакета в поток необходимо также записать в него нули в размере недостающей до 2920 байт части. Только после превышения порога байт сообщение будет отправлено. Формат заголовка пакета указан в таблице 7.

В случае ответа на команду в 3-м байте будет указан соответствующий команде код. Помимо ответов, в режиме набора данных, DANTE будет посылать в одностороннем режиме пакеты, содержащие набранные события. В таких пакетах 4-й байт будет всегда нулевым, а 3-й байт будет иметь значение в соответствии с режимом набора:

- 2 - для набора только спектра
- 3 - для набора кадров
- 4 - для набора событий

Байт	Значение	Описание
0	0xAA	Спецсимвол начала пакета
1	0xEE	Спецсимвол начала пакета
2	0xD + CMD(4 bits)	Код команды
3	Board	Номер платы
4	Packet number	Номер пакета. Такой же, как и в команде
5	Length (High byte)	Размер бинарной части (старший байт)
6	Length (Middle byte)	Размер бинарной части (средний байт)
7	Length (Low byte)	Размер бинарной части (младший байт)

Таблица 7 — Формат заголовка ответного пакета DANTE.

– 6 - для вывода мониторных данных

Байты 5, 6, 7 соответствуют размеру бинарной части пакета (в количестве 32-битных слов)

Формат также подразумевает оборачивание посылаемых сообщений открывающей и замыкающей 2-байтовыми последовательностями имеющими значения 0xDD 0xAA и 0xDD 0x55 соответственно. Если в сообщении имеются байты, совпадающие с одним из байтов открывающей последовательности - они должны быть экранированы с помощью добавления перед каждым из них байта со значением 0xDD.

Пакеты, приходящие с устройства, в отличии от команд, не оборачиваются в открывающие и замыкающие последовательности спецсимволов. Также, к самому пакету не применяется никаких правил экранирования содержимого.

Настройка платы

После соединения с DANTE и перед запуском набора измерений необходимо провести конфигурацию плат. Для этого в инструкции по разработке платы предоставлена точная последовательность команд, необходимых для установки каждого параметра набора. Эту последовательность необходимо выполнить для всех плат DANTE. В процессе настройки будут заданы параметры оцифровки сигнала, такие как инвертирование, время восстановления, режим преобразова-

Слово	Биты	Содержание
1	31-18	Время прихода события (14 бит) в 8 нс бинах от начала набора. Младшие биты
1	17-16	зарезервировано
1	15-0	Амплитуда события (16 бит)
2	31-30	Контрольные биты. Для событий значение равно "00"
2	29-0	Время прихода события (30 бит) в 8 нс бинах от начала набора. Младшие биты

Таблица 8 — Формат 32-битного слова в бинарной части пакета Dante.

ния амплитуд, порог, время формирования сигнала, порог наложения, и т.п. Описанная в мануале последовательность, помимо непосредственно задания параметров описывает способ их кодировки в значения регистров. Команды по заданию параметров также «разбавляются» командами, задающими необходимые программные константные значения битов регистров. Последовательность команд довольно длинная и малопонятная, поэтому ее описание будет опущено.

Набор амплитуд событий

В начале тестирования DANTE было принято решение сперва отладить процесс набора с оцифровкой сигнала в амплитуды с помощью встроенных в устройство алгоритмов. Для начала набора точки в таком режиме необходимо выполнить следующую последовательность действий:

1. Записать в регистр ACQUISITION_TIME необходимое время набора
2. Записать в регистры MAP_POINTS и TIME_PER_MAP_POINT параметры вывода статистики для мониторинга
3. Записать в регистр ACQUISITION_STATUS бит, соответствующий старту набора

После выполнения этих действий, DANTE в одностороннем режиме начнет посылать сообщения содержащие параметры набранных событий и сообщения, содержащие статистику для онлайн мониторинга набора.

Каждое событие в бинарной части закодировано двумя 32-битными словами в формате, описанном в таблице 8.

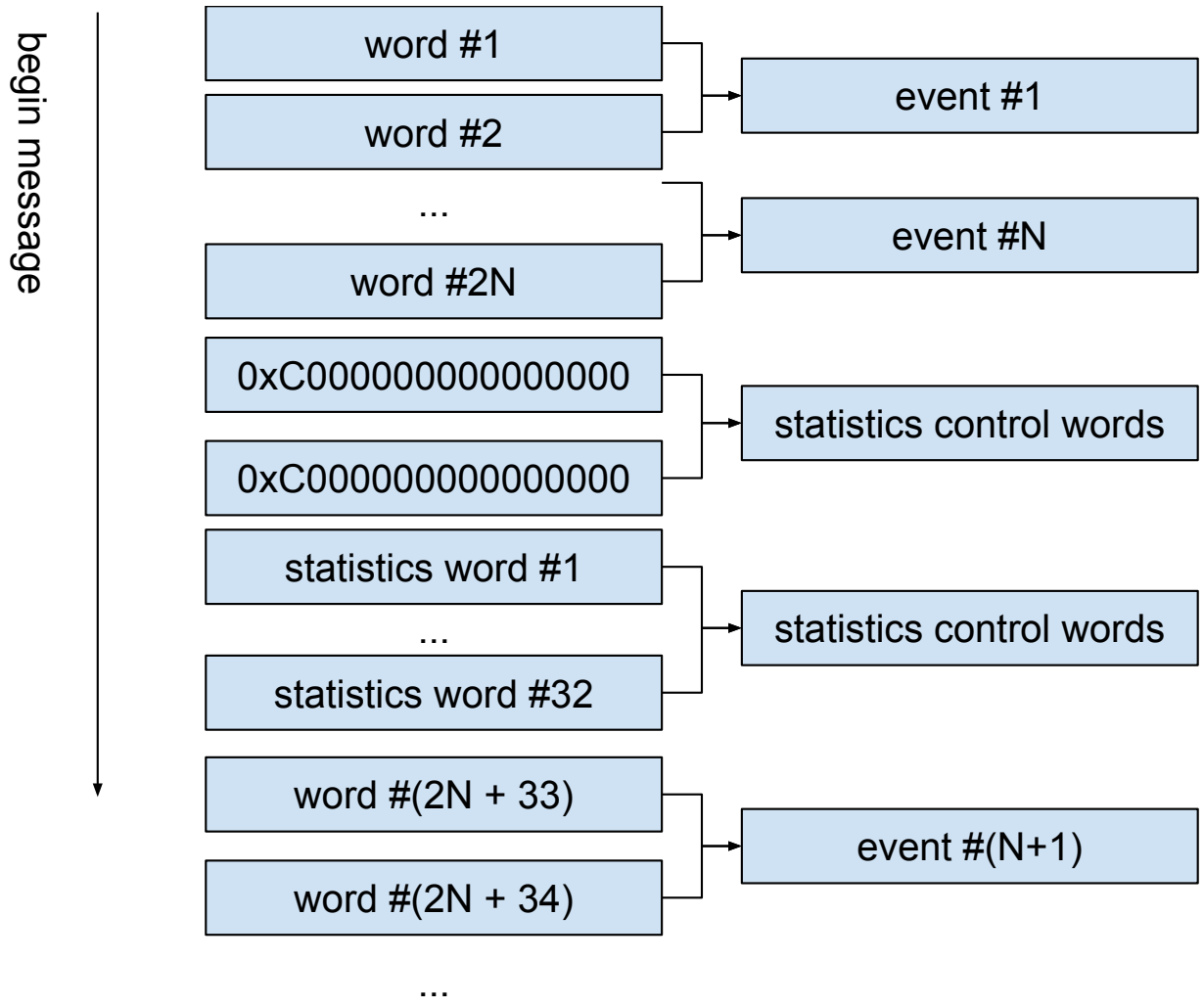

Рисунок 4.8 — Формат сообщения DANTE.

Части, содержащие статистические данные, имеют фиксированную длину в 32 слова и отделяются от событий 2-мя словами, имеющими контрольные биты со значениями «11», т.е. равные "0xC000000000000000". Т.к. DANTE подключается в отладочном режиме, в процессе обработки из бинарных данных сообщения со статистикой просто пропускаются, поэтому описание формата будет опущено.

На рисунке 4.8 изображена схема бинарного формата DANTE.

4.2.2 Разработка сервиса

При разработке сервиса для набора событий с помощью DANTE было решено использовать режим управления через TCP/IP. На основе модуля

python-df-tcp был разработан проект DANTE-server, исходный код которого расположен в репозитории [47].

Для протокола сообщений DANTE был написан парсер, код которого находится в файле protocol.py в корне проекта. Парсер может создавать сообщения с командами (экранирование предусмотрено) и извлекать пакеты из потока бинарных данных.

Файл client.py, расположенный в корне проекта реализует функции по подключению к сокетам DANTE, считыванию и склейке пакетов, поступающих от устройства. Код написан с использованием стандартной библиотеки Python 3 asyncio. Скрипт создает 2 очереди:

- SEND_QUEUE - используется для передачи команд на DANTE
- GET_QUEUE - используется для считывания склеенных пакетов, приходящих с DANTE

Сам сервис, исходный код которого расположен в файле redirecter.py реализует функции:

1. Инициализации плат DANTE. Инициализация происходит после получения от управляющего компьютера пакета содержащего команду типа "init". После получения команды сервер последовательно выполняет набор действий, описанных в руководстве по устройству DANTE для каждой из 7 подплат.
2. Набор точки. Начало набора происходит через запись в описанные в подглаве 4.2.1 регистры значений, соответствующих желаемым параметрам набора. После этого программа ожидает приема пакетов. Время ожидания равно времени набора плюс константа. Во время ожидания сервер принимает пакеты, вырезает из них статистические данные и извлекает набранные события. По истечении ожидания сервер считает набор завершенным, компонует полученные события в пакет dataforge-envelope и отправляет их в виде ответа на управляющий компьютер.

Также было разработано виртуальное устройство DANTE, которое работает в соответствии с описанном в технической документации поведением. Исходный код виртуального устройства находится в папке test в файле server.py.

Работоспособность через API

После создания сервиса DANTE, взаимодействующего с устройством через TCP/IP началось тестирование работоспособности такого решения. Для этого проводился набор и анализ данных фоновых событий установки «Троицк ню-масс». В процессе тестирования программа провела успешную инициализацию плат DANTE и смогла запустить набор событий. Однако набранные файлы, как оказалось, имели нереалистично низкие и плавающие скорости счета. Т.к. установка «Троицк ню-масс» в процессе набора файлов находилась в относительно стабильном режиме, такие эффекты не могли быть вызваны физикой процессов установки. Описанные проблемы не позволяли начинать набор статистики с помощью DANTE. Появилась необходимость нахождения причин и устранения ошибок.

Анализ работы разработанного сервера и набранных данных показал следующие моменты:

1. DANTE перестает работать корректно после переподключения к устройству.
2. Если в процессе набора событий приходит команда - устройство зависает. Это исключает возможность проверки завершения набора с помощью регистров.
3. Время события по видимому записывается не в единицах 8 нс.
4. Иногда происходит обрезание пакетов (последнее событие пакета имеет размер меньше двух слов)
5. Иногда DANTE начинает присылать пустые события.
6. Некоторые амплитуды событий имели некорректные значения (больше 16 бит).

Для понимания причин такого поведения сперва было решено перепроверить правильность установки конфигурации при инициализации плат. В комплекте с DANTE шло приложение, позволяющее проводить набор данных в ручном режиме через графический интерфейс. Набранные при работе через эту программу файлы выглядели явно лучше чем набранные с помощью сервиса: скорость счета была значительно выше, стабильнее и ближе к ожидаемым значениям. К счастью, через интерфейс программы можно было задать протокол управления: TCP/IP, либо USB. Для проверки корректности работы было реше-

но провести с помощью программы инициализацию плат с помощью TCP/IP и с заранее известными параметрами и с помощью программы Wireshark записать весь трафик, передаваемый на сокеты DANTE. Затем, анализируя трафик предполагалось вычленить из него команды и ответы на них и сверить очередность и соответствие ответов с процессом инициализации через разработанный сервис. В процессе анализа трафика оказалось что в работе сервиса действительно присутствовали некоторые ошибки, однако после их устранения желаемой работы устройства все равно не удалось достигнуть. При абсолютно одинаковой последовательности инициализации плат и старта набора данные, набранные разными способами различались.

Вторым этапом была предпринята попытка проведения набора событий с помощью DANTE через порт USB. Т.к. идущая в комплекте программа использует именно USB при наборе событий это решение показалось разумным. С помощью стандартного модуля Python ctypes был создан скрипт, проводящий процесс инициализации с помощью драйверов DANTE. Однако, при выполнении, скрипт зависал на начальных командах. Возможно это было связано с тем, что поставляемая библиотека общения с драйвером была устаревшая. Т.к. предположительно стандартная программа набора для DANTE была написана на LabView такой вариант выглядит реалистичным. Попытки комбинирования протоколов (Т.е. использование TCP/IP для инициализации и USB для набора событий) также не сработало, т.к. устройство, как оказалось, без перезагрузки не может работать одновременно с двумя протоколами.

Работа через стандартную программу

В условиях сильной ограниченности по времени было решено отказаться от работы с устройством непосредственно через API и создать новый сервис, который будет привязан к стандартному приложению DANTE и проводить набор с помощью эмуляции кликов мыши по графическому интерфейсу (кликер). Это очень не рекомендуемый подход, который имеет очевидные недостатки (привязанность к конкретному разрешению экрана, хрупкость системы, невозможность использовать компьютер во время набора), однако с учетом попыток

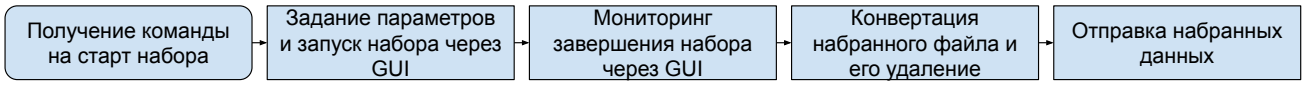

Рисунок 4.9 — Схема работы кликера.

разбора протоколов взаимодействия такой подход оказался единственным работоспособным вариантом.

Кликер был также написан на Python 3 на основе модуля python-df-tcp. В качестве фреймворка, эмулирующего нажатие мыши был использован модуль PyAutoIt, предоставляющий обертки функций AutoIt[48]. Схема набора событий с помощью кликера изображена на 4.9.

Помимо конфигурации набора, описанного ранее здесь также присутствует этап конвертации данных, полученных стандартной программой в dataforge-envelope формат. Отправляемый сервисом пакет с набранными событиями по структуре очень похож на формат обработанного файла Лан10-12PCI. Бинарные данные пакета имеют одинаковый формат в обоих случаях, метаданные DANTE выглядят так:

```
{ "start_time": start_time,
"external_meta": {/*...*/},
"acquisition_time": 30,
"type": "reply",
"reply_type": "aquired_point",
"status": "ok",
"time_coeff": 8,
"total_events": 1000,
"device": "DANTE daisy chain rev 1.8",
"dpp_params": {/*...*/}}
```

Листинг 4.2 — Метаданные пакета с набранными DANTE событиями.

Из отличий от Лан10-12PCI здесь: параметры устройства вынесены из тега "external_meta"в корень метаданных в тег "dpp_params добавлено поле "device содержащее информацию об устройстве, с помощью которого проводился набор.

4.2.3 Результаты тестирования платы

В процессе интеграции прототипа детектора «KATRIN» с системой считывающей электроники DANTE были выявлены следующие проблемы:

- Работа в режиме взаимодействия через TCP/IP явно неправильна. Как оказалось, в процессе набора пакеты часто обрезаются и не все события доходят до клиента, а скорость счета сильно изменяется при наборе данных в одинаковых условиях: такая величина изменения не может быть объяснена физическими процессами. Также суммарный объем передаваемых данных, при анализе через Wireshark, оказался намного меньше объема данных, полученных при наборе с помощью стандартных средств, идущих в поставке с DANTE. Это говорит о том, что ошибка передачи событий происходит внутри самого устройства и не может быть объяснена ошибками парсинга, связанными с обработкой битых пакетов. В дополнение неясно использование 4 сокетов - не очевидно, что это может максимизировать пропускную способность с учетом того, что к устройству можно подсоединить только один ethernet кабель.
- Поставляемые с DANTE драйвера для работы с USB по видимому устаревшие. В процессе их тестирования не удалось добиться инициализации подплат - вызов функций библиотеки драйвера приводит к зависанию программы. Этот и предыдущий пункты приводят к необходимости использования стандартной программы для взаимодействия с DANTE. При работе на установке Троицк-ню масс для обхода этих проблем был использован кликер - такое решение подходит для тестирования, однако на реальных сеансах «KATRIN», особенно с учетом высоких требований к скоростям счета, такой метод неприемлем.
- Работа с устройством по API требует задания низкоуровневых настроек через регистры, кодирования параметров набора в приемлемые для них форматы и задания «магических» констант. Такое задание явно не удобно и, с учетом потенциальной стоимости DANTE, может быть легко и без сильного удорожания решено добавлением к устройству микрокомпьютера, который будет производить конвертацию внутренних команд и ответов DANTE в высокоуровневый и понятный формат.

Глава 5. Обработка непрерывного сигнала

При переходе на поиски стерильного нейтрино в диапазоне энергий до 5 кэВ появляется проблема обработки высоких скоростей счета детектора [5]:

– Численное моделирование искажений спектра в зависимости от параметров систематики выявило главные источники, в числе которых оказалось мертвое время электроники. Кроме того важна точность определения мертвого времени.
– Поиск сигнатуры стерильного нейтрино требует большой статистики. Согласно предложениям, для эффективной работы система должна справляться со скоростями порядка 40-50 кГц. MADC имеет мертвое время 7.2 мкс и не способен справится с такими скоростями. На графике 5.1 показана эффективность выделения событий в зависимости от скорости счета для нескольких значений мертвого времени.
– Детектор, в текущей конфигурации системы сбора данных генерирует сигналы с выбросом размером около 15 мкс. При высоких скоростях счета выброс будет искажать амплитудный спектр.

Было решено провести обновление аппаратной части, заменить старую аппаратную оцифровку на плату Лан10-12PCI, набирать с помощью нее квазинепрерывный сигнал и программно выделять из него параметры событий. Принцип набора данных аналогичен [49]. При таком подходе скорость счета определяется качеством разделения наложенных событий.

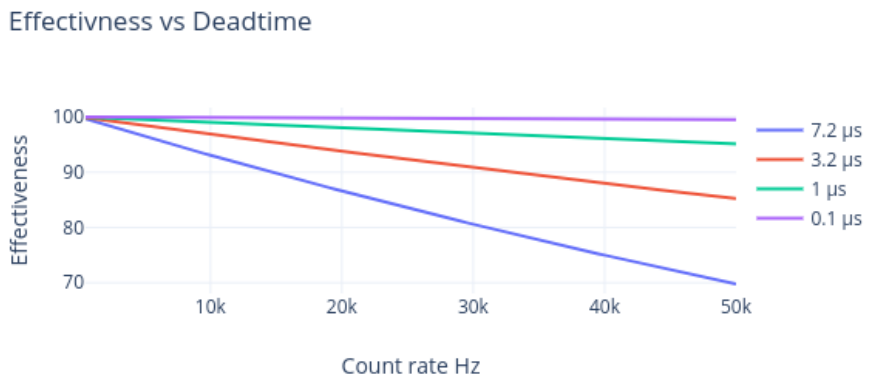

Рисунок 5.1 — Эффективность выделения событий в зависимости от скорости счета.

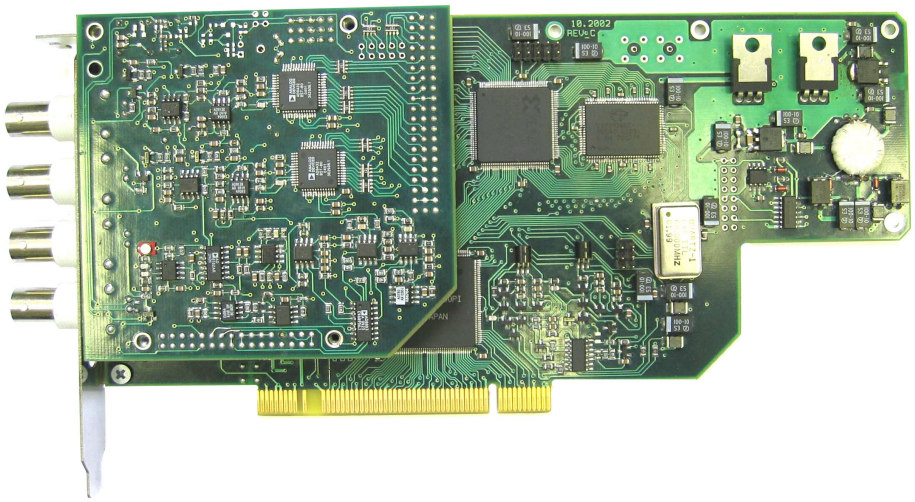

Рисунок 5.2 — Плата Лан10-12PCI.

На момент разработки существовало несколько способов разделения. Например, коррекция амплитудного спектра с помощью нейронный сетей[50]. Алгоритм использует смоделированные спектры в качестве обучающей выборки, а затем использует полученную сеть для анализа данных. Недостаток этого подхода заключается в том, что он требует знания формы исходного неискаженного спектра амплитуд для правильной процедуры обучения. Также любые аномалии в распределении времени события могут привести к неверным результатам коррекции.

Другой способ описан в [51]. Он основан на аналитической модели, которая предсказывает мертвое время и искажения спектра, вызванные наложениями сигналов. Несмотря на хорошие результаты, модель жестко привязана к аналитической формуле импульса и поэтому неприменима в других системах без существенных изменений.

Предлагаемые нами алгоритмы коррекции также основаны на форме отдельного события. Однако вместо использования априорной информации, такой как форма спектра или формула аналитического сигнала, он опирается только на данные непрерывного сигнала, собранные с помощью АЦП, и извлекает все необходимые параметры из них. Это делает алгоритмы применимыми для различных задач оцифровки сигнала без существенных изменений.

5.1 Характеристики Лан10-12PCI

Плата произведена ЗАО "Руднев-Шиляев" и представляет собой встраиваемый в ПК через PCI 12-битный АЦП. Лан10-12PCI имеет 2 аналоговых входа, канал внешней синхронизации и канал внешней тактовой частоты. Возможные частоты дискретизации составляют от 100 МГц до 6.1035 кГц или любая частота в диапазоне от 5 МГц до 100 МГц при работе с внешней тактовой частотой. Возможные длины - от 28 до 220 бинов на кадр. Имеется внутренний буфер размером 512 кб, который может одновременно хранить один кадр вне зависимости от его размера. Запись кадров может проводиться в трех режимах

1. По превышению порога
2. По событию канала внешней синхронизации
3. По программному триггеру (через API)

Подробные характеристики можно найти в [52].

Плата была выбрана в основном из-за ее низкой цены, однако для нашей задачи она имеет некоторые преимущества. Это:

1. Наличие относительно большого внутреннего буфера и возможность ручного считывания кадра по программному триггеру.
2. Наличие комплектных драйверов для Windows и Linux и полноценного API к ним.

Перед интеграцией платы было проведено тестирование набора кадров в режиме записи по превышению порога. Результаты представлены в таблице 9.

Тестирование показало что даже при минимальном размере кадра, эффективная скорость счета составляет примерно 3-4 кГц, что совершенно не соответствует требованиям для системы сбора данных установки. Дальнейшее тестирование показало, что основной вклад в мертвое время дает сброс данных из внутреннего буфера платы на ПК. Т.о. существует возможность оптимизации набора с помощью минимизации количества сбросов буфера платы.

Размер кадра - 1 мкс

Частота, кГц	Эффективная частота, кГц	Эффективность
40	2,553	0,064
20	2,554	0,128
5	2,558	0,512
1,25	1,242	0,993

Размер кадра - 2 мкс

Частота, кГц	Эффективная частота, кГц	Эффективность
40	2,489	0,062
20	2,537	0,127
10	2,527	0,253
1,25	1,238	0,99

Размер кадра - 4 мкс

Частота, кГц	Эффективная частота, кГц	Эффективность
40	0,464	0,012
10	0,918	0,092
2,5	0,788	0,315
0,313	0,31	0,992

Таблица 9 — Эффективность Лан10-12PCI при наборе по триггеру

5.2 Набор непрерывного сигнала

Для минимизации количества сбросов мы выставляем максимальную длину кадра в 2^{20} и проводим непрерывный набор по программному триггеру, который генерируется сразу после считывания предыдущего кадра из буфера АЦП. Т.о. с платы считываются длинные кадры (размер кадра намного больше размера события) без привязки к событиям. На рисунке 5.3 представлена описанная схема набора.

На выходе мы имеем практически непрерывную сигнал с разрывами только на сброс данных, которые заранее определены и могут быть учтены при обработке как живое время. Т.о. данные не содержат искажений, обусловленных аппаратным мертвым временем. Извлечение параметров событий перекладывается с аппаратной составляющей платы, которая в нашем случае сильно ограниченна, на ПК. Т.о. скорость обработки ограничивается только парамет-

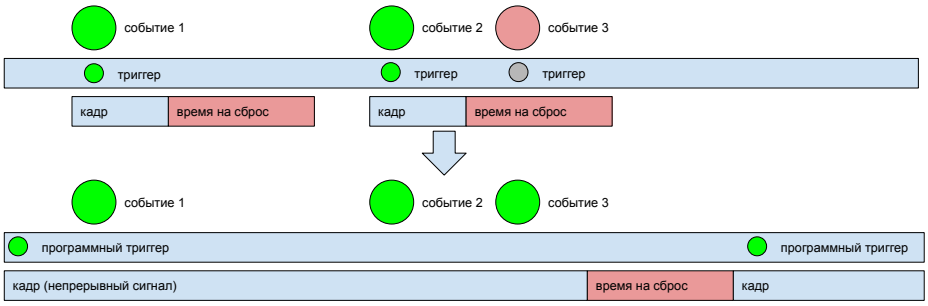

Рисунок 5.3 — Набор кадров по триггеру (сверху) и квазинепрерывного сигнала по программному триггеру (снизу).

Частота оцифровки, Гц	мертвое время за 10 с набора, с	кол-во кадров	время на запись кадра, с	Эффективность записи, %
1.25e+7	3.96	72	0.055	0.604
6.25e+6	2.45	45	0.054	0.755
3.125e+6	1.28	26	0.049	0.872

Таблица 10 — Эффективность Лан10-12PCI при наборе квазинепрерывного сигнала

рами компьютера (также обработка может быть отложенной, если ресурсы не позволяют делать ее в реальном времени).

Для Лан10-12PCI описанный метод набора был исследован на некоторых частотах оцифровки. Параметры живого времени для разных частот дискретизации приведены в таблице 10. Видно, что операция переноса занимает одинаковое время для любого размера кадра. Для набора экспериментальных данных была выбрана частота оцифровки - 3.125e+6 Гц (320 нс). При такой частоте плата обеспечивает 87% живого времени и дает оптимальное соотношение размеров набираемых файлов и спектрального разрешения.

5.2.1 Предобработка данных

Для экономии места жесткого диска, из набираемых кадров, содержащих непрерывный сигнал, вырезаются только интервалы, содержащие события. Обрезание производится с помощью алгоритма zero-suppression. Алгоритм фильтрует все бины, имеющие амплитуду ниже порога и не имеющие соседних

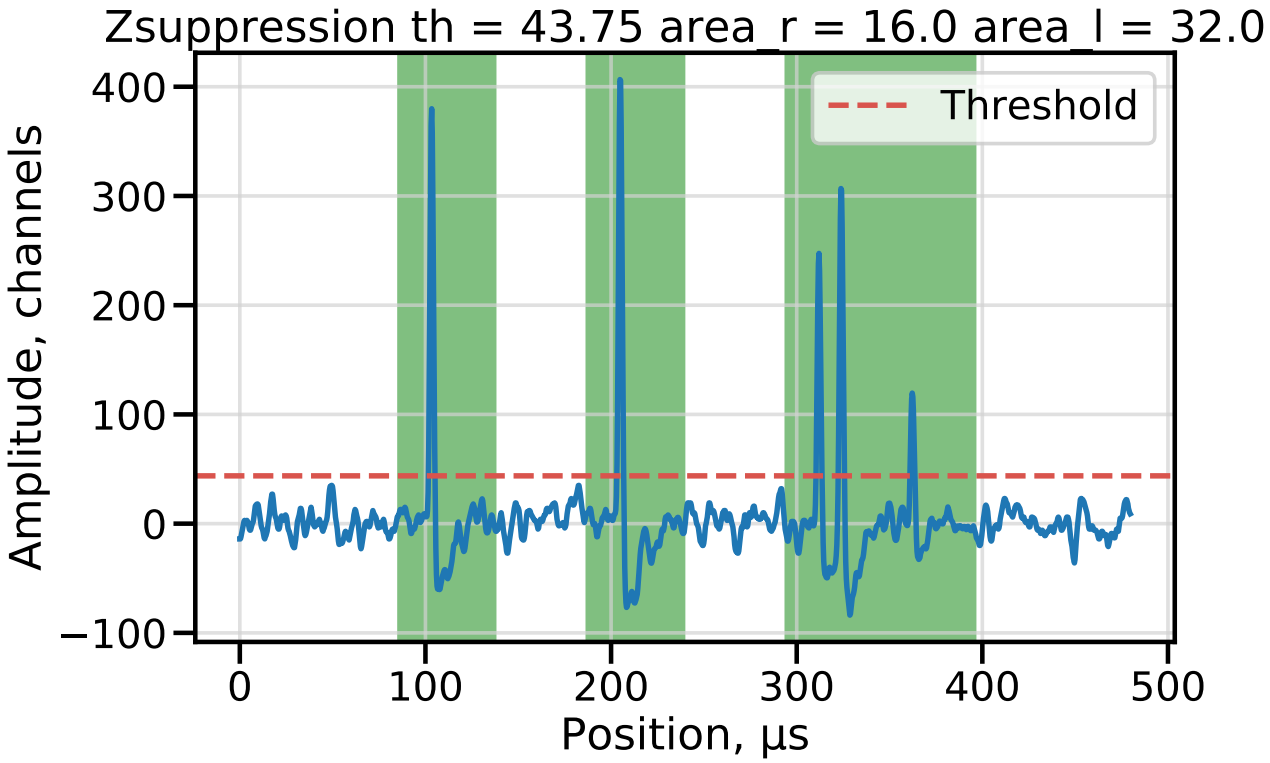

Рисунок 5.4 — Алгоритм zero-suppression.

бинов выше порога в заданной окрестности, вырезаются. Т.о. сигнал разбивается на кадры переменной длины, содержащие события.

Алгоритм операции zero-suppression:

1. Создание маски
    а) Бинаризация кадра по пороговому значению
    б) Применение морфологической диляции с ядром, соответствующем желаемым границам события
2. Вырезание отфильтрованных участков по созданной маске

Данная реализация позволяет избежать дублирования пограничных областей событий за счет слияния близких событий в один подкадр. Далее в работе для простоты мы будем ссылаться на подкадр, полученный в результате предобработки алгоритмом zero-suppression, как простой «кадр» и «кадр непрерывного сигнала» - для исходного кадра Лан10-12PCI до предобработки.

5.3 Симуляция непрерывного сигнала

Для разработки и тестирования алгоритмов выделения параметров событий требуется большая промаркированная выборка кадров непрерывного сигнала. Единственным реальным способом ее получения является моделирование. В ходе работы мы разработали генератор данных платы Лан10-12PCI, который создает непрерывные кадры сигнала, имитирующие вывод реального устройства. Для создания события в кадре генератор использует два определяющих параметра - амплитуду и время. Поскольку параметры определены явно, кадры непрерывного сигнала создаются уже размеченными. Вывод генератора рассчитан на полную совместимость с выводом реальной платы. Такой подход к генератору имеет свои преимущества и недостатки. С одной стороны, сырой вывод платы является первичным входом для всей системы обработки, использование таких сгенерированных размеченных данных позволяет протестировать все этапы обработки. Кроме того, нет необходимости разрабатывать дополнительный код тестирования (все манипуляции с данными будут такие же как и для реальных измерений). Еще одним преимуществом является возможность разработки без самой платы, поскольку генератор также можно использовать в качестве виртуального устройства. Недостатком является необходимость выполнения всех этапов обработки для тестирования любой компоненты. Это значительно снижает скорость тестирования (особенно для конечных этапов обработки) по сравнению со случаем генерации данных для конкретного этапа. Также может появиться узкое место

в случае, если некоторые промежуточные этапы занимают намного больше времени, чем тестируемый.

Архитектурно, наш генератор данных состоит из двух модулей:

– генератор шума,

– генератор форм событий.

Непрерывный сигнал получается путем наложения чистых форм событий на шумовой фон.

5.3.1 Моделирование шумового фона

Мы предполагаем, что шум состоит из большого количества низкоамплитудных событий, равномерно распределенных по времени и имеющих ту же форму, что и физическое событие. Таким образом, шум может быть создан путем явного наложения форм, созданных чистым генератором форм событий без дополнительных модулей. Однако, чтобы ускорить симуляцию, можно оптимизировать генерацию шума.

Принимая во внимание высокую частоту и распределение пиков фоновых событий, можно генерировать конечную форму наложенных шумовых событий с помощью метода, основанного на применении композиции операций сглаживания к массиву случайных чисел.

Алгоритм генерации шума тогда работает следующим образом:

1. Создание массива случайных чисел с равномерным распределением. Размер массива равен размеру выходного кадра.
2. Размытие массива с большим константин ядром для достижения среднего числа пиков, аналогичных реальному шуму. Грубая подстройка распределения амплитуд шума.
3. Двойное размытие массива с небольшими константными ядрами для достижения необходимого размытия небольших пиков. Тонкая подстройка распределения амплитуд шума.
4. Округление значений массива до целого числа и сдвиг их значений на 2 бита, для обеспечения соответствия 12-битным значениям полученным реальной платой.

Этот алгоритм позволяет генерировать с небольшими затратами суммарную форму от большого количества равномерно и плотно распределенных по времени событий. В отличие от метода наложения событий, результирующий кадр не будет размечен, что приемлемо для шума.

Для проверки генератора шума, мы сравнили смоделированный шум с реальным. Настоящий шум был получен путем комбинированием небольших фрагментов, оставленных алгоритмом zero-suppression до события. Размеры ядра свертки были подобраны вручную. При подборе мы наблюдали за:

1. Соответствием амплитудных спектров реального и генерируемого шума.

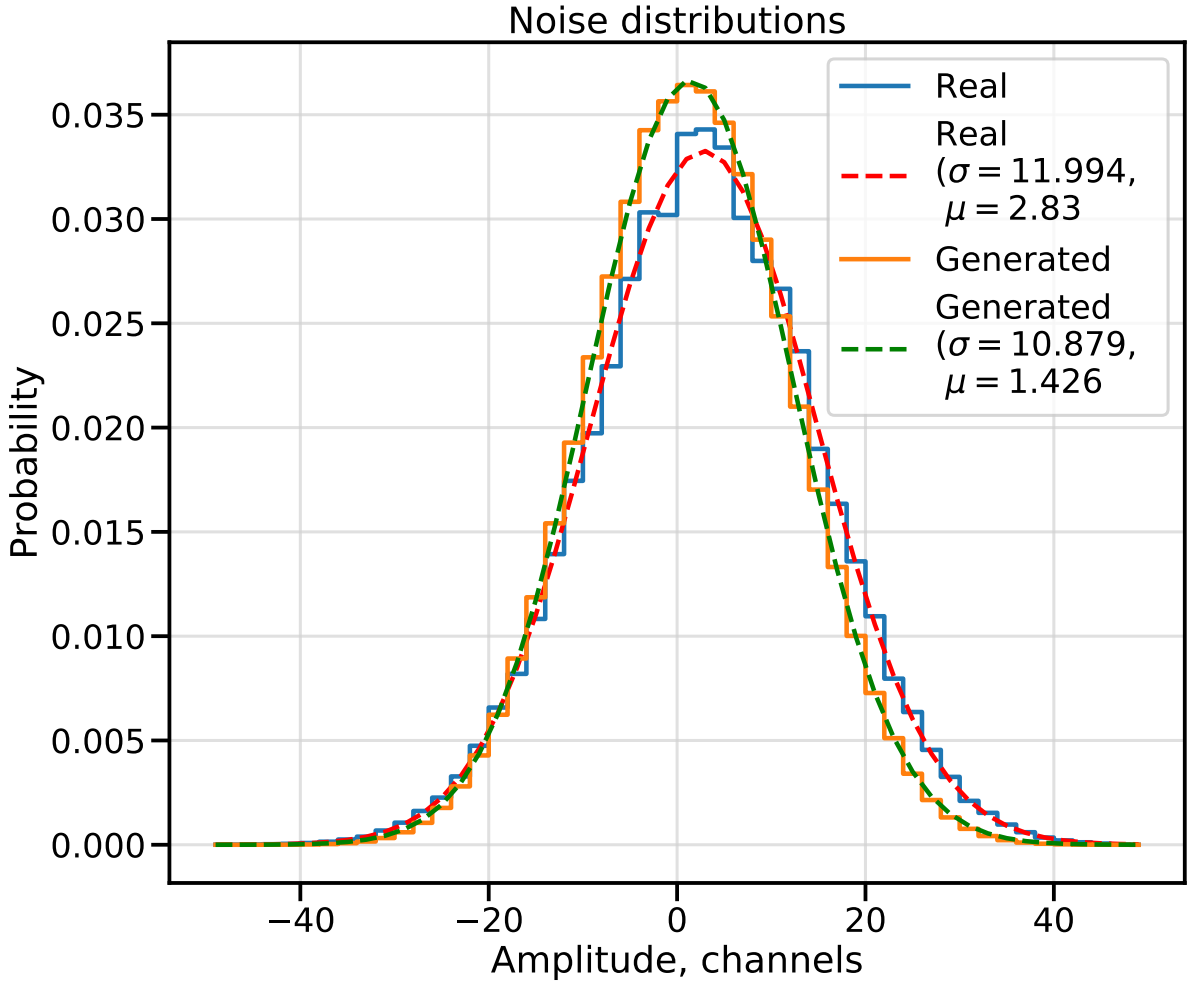

Рисунок 5.5 — Сравнение шумовых спектров.

2. Визуальным совпадением формы шумовых пиков и средних расстояний между соседними пиками для некоторых случайно выбранных фрагментов.

Как видно по изображенным на 5.5 распределениям базовая линия может колебаться между сериями измерений. Тестирование проводилось на более поздних наборах данных и мат. ожидание для шумов сместилось на примерно на 12 каналов. Однако статистические параметры шумов находятся близко друг к другу. Далее будут приведены результаты тесты выделения событий с различными значениями генератора шумов.

Также теперь известны мат. ожидание и среднеквадратичное отклонение для распределения амплитуд шумов, которое понадобится нам при анализе формы.

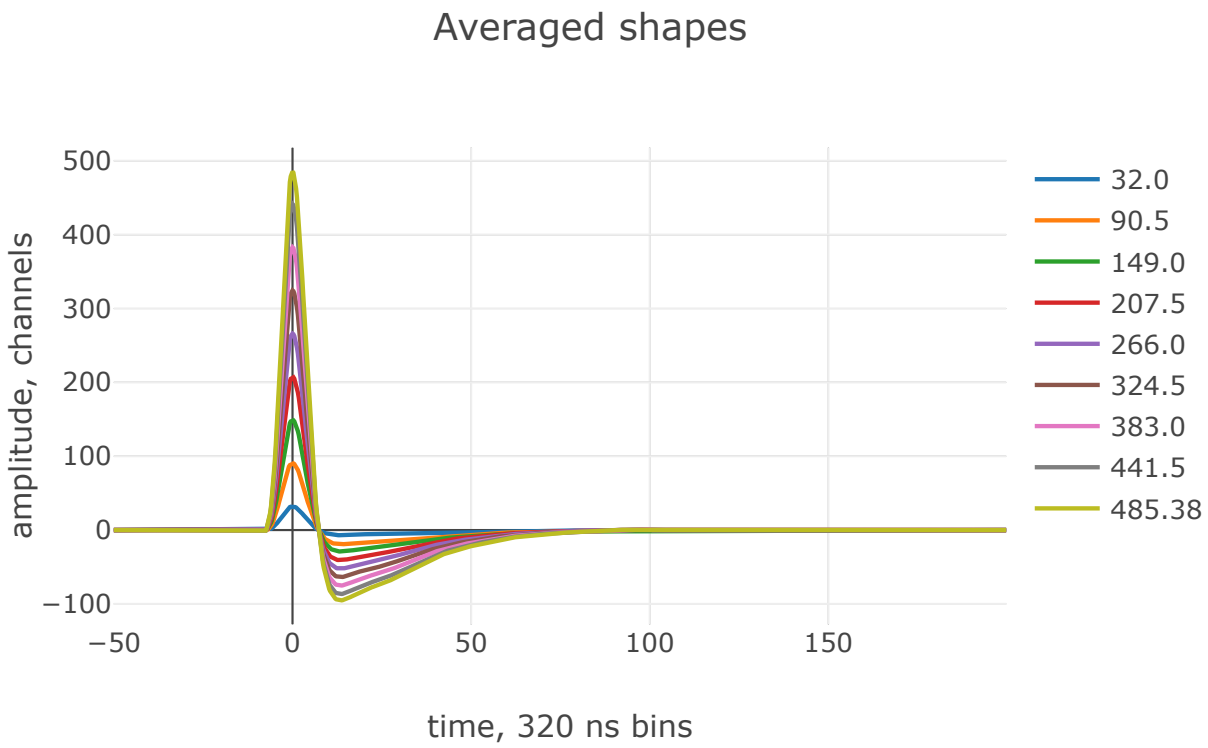

Рисунок 5.6 — Усредненные формы событий.

5.3.2 Моделирование формы события

Чтобы смоделировать событие, необходимо знать его реальную, не искаженную шумами форму. Формы событий могут быть извлечены из данных путем группировки событий по значению амплитуды и последующего усреднения форм по группам. Размер кадра после обрезки также дает предварительное условие, которое может отфильтровывать кадры с наложениями.

Результаты усреднения событий по группам показаны на рисунке 5.6. По внешнему виду форм видны следующие особенности:

1. Пик события хорошо описывается гауссианом.
2. Выброс после события после достижения максимума экспоненциально затухает.
3. Соотношение между амплитудами пика и выброса постоянно.
4. Также постоянно расстояние между экстремумами пика и выброса.

Для нашей конкретной формы была подобрана аналитическая кусочно-заданная функция 5.1.

$$
\begin{aligned}
&t_f = 2.9604, p = 2.2056, t_a = 0.3701,\\
&\sigma = 0.3416, f_b = 3.125e+6,\\
&g(x) = exp(-\frac{|\sigma f_b x|^p}{2}), g_r(y) = \frac{(-2log(y))^{1/p}}{\sigma f_b},\\
&s(x) = \begin{cases} (\frac{1}{(1+2xf_bs)^{t_f}} - 1)exp(-xf_bs), x > 0 \\ 0, x \leqslant 0 \end{cases},\\
&S(x,a,o) = (g(x-p) + s(x - g_r(0.1) - p)t_a)a.
\end{aligned}
\tag{5.1}
$$

Здесь $g(x)$- Гауссова функция, $g_r(y)$ - обратная Гауссова функция (только правая часть), $s(x)$ - функция, описывающая выброс. Константы подобраны фитированием. Объясним формулу. Начало и пик и неполная часть спада события описываются только функцией Гаусса, т.е $g(x)$. На спаде, в месте, где амплитуда переходит порог равный 10 % от максимальной амплитуды, к Гауссиану добавляется хвост, выраженный функцией $s(x)$. Для вычисления пороговой точки используется обратный Гауссиан $g_r(y)$. Наконец, переменные a и p в формуле $S(x,a,p)$ задают амплитуду (задается линейно) и отступ от начала кадра соответственно.

На рисунке 5.7 показаны результаты фита функции события на усредненную форму для нескольких амплитуд.Видно, что подобранная функция хорошо описывает реальное событие.

5.3.3 Валидация генератора

Для проверки соответствия реальных и полученных генератором кадров были построены распределения квадратичных отклонений форм восстановленных событий. Для этого для двух типов были созданы выборки кадров, содержащих по одному событию (в реальных данных для фильтрации одиночных событий использовался порог по длине кадра). Из кадра выборки затем извлекались параметры события. Для получения амплитуды и положения пика использовалась полиномиальная аппроксимация степени 2 по 8 точкам в окрестности максимума кадра. После этого по формуле 5.2 вычислялось отклонение в

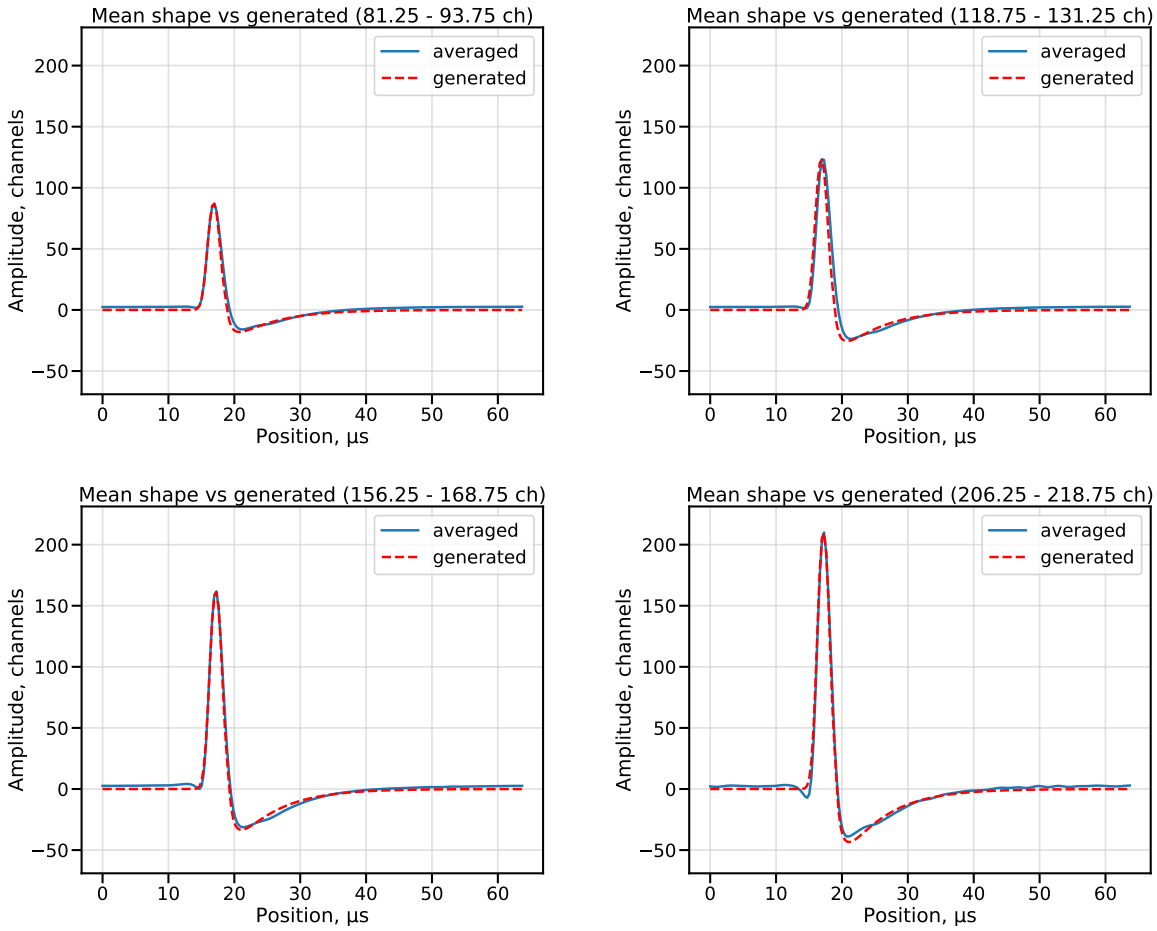

Рисунок 5.7 — Сравнение аналитической у реальной усредненной форм для разных амплитуд.

кадре χ_e^2 ($p(n)$ - амплитуда бина n в кадре, $p_r(n)$ - амплитуда восстановленного события в том же бине, σ - среднеквадратичное отклонение, взятое из распределения амплитуд реального шума, полученного при создании генератора шума).

$$\chi_e^2 = \sum_0^n \frac{(p(n) - p_r(n))^2}{\sigma} \tag{5.2}$$

По полученным χ_e^2 был построен спектр, изображенный на рисунке 5.8.

По графикам видно, что:

1. Распределения по форме совпадают с хи-квадрат распределением.
2. Среднее отклонение для обоих наборов данных в пределах десятых равно единице.

Таким образом, по результатам валидации, можно утверждать, что наша модель генератора достаточно точно описывает реальные данные и может быть использован для создания размеченных выборок кадров непрерывного сигнала для тестирования алгоритмов выделения.

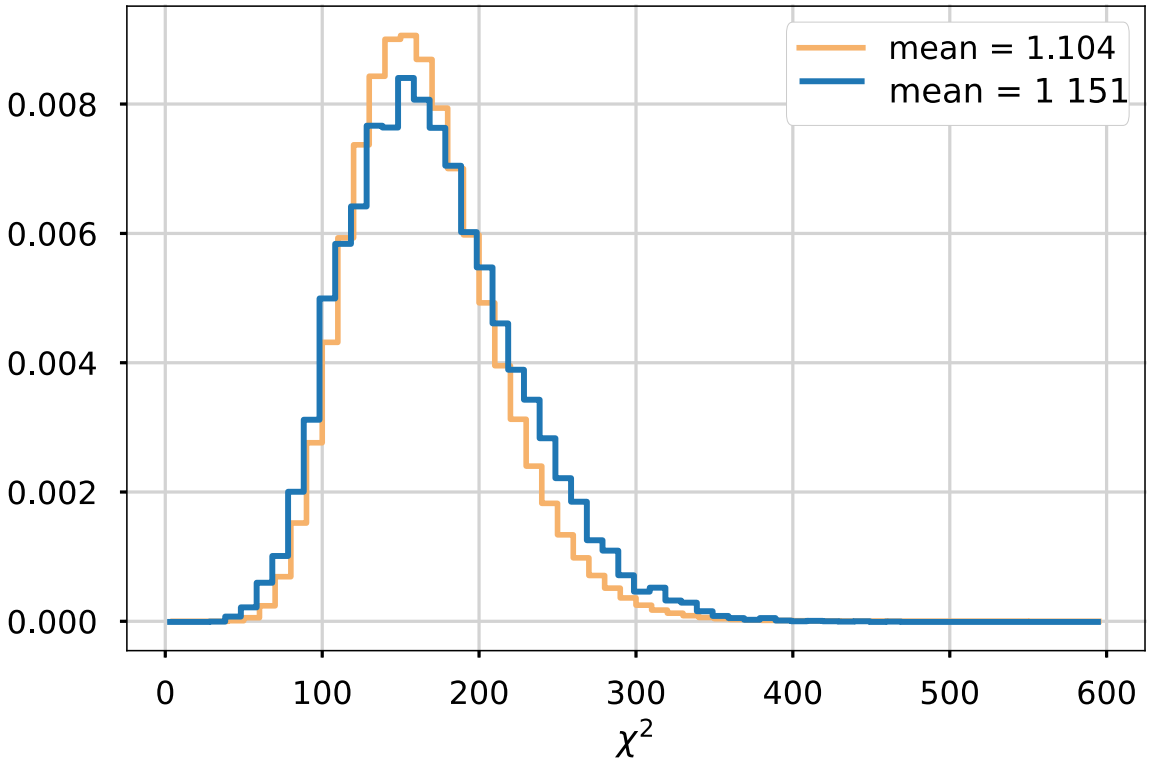

Рисунок 5.8 — Спектр отклонений χ^2.

5.3.4 Алгоритмы выделения параметров событий из кадра

В этой подглаве будут описаны три разработанных алгоритма выделения параметров событий. С помощью алгоритмов будут обработаны тестовые выборки, по результатам будет оценено качество работы. Тестовые выборки являются симуляцией набора с Лан10-12PCI со скоростью счета 40 кГц (которая соответствует максимальной скорости счета, ожидаемой в «Троицк ню-масс») и продолжительностью набора 35 с (тестовые выборки не идентичны друг другу, но аналогичны по параметрам).

Определим эффективную скорость счета как:

$$\tau = \frac{T}{N_0}\left(1 - \frac{N}{N_0}\right), \tag{5.3}$$

где T- полное время набора, N- количество распознанных событий, N_0- реальное количество событий в кадре.

В качестве метрик качества обработки будут выступать:

1. скорость обработки,

2. процент распознанных событий,
3. процент ложноположительных срабатываний,
4. эффективная скорость счета, определяемая по формуле 5.3.

Для определения соответствий между исходными реконструированными событиями используются следующие правила:

1. Реконструированное событие соответствует исходному, если:
    а) текущее реконструированное событием ближайшее к исходному по сравнению с остальными реконструированными событиями и
    б) разница между положениями пика не превышает 10 бинов (т.е. 3.2 мкс).
2. Исходное событие считается наложенным, если:
    а) разница между положениями пика исходного и реконструированного событий не превышает 10 бинов и
    б) соответствующее реконструированное событие имеет другое более близкое исходное событие.
3. Исходное событие считается нераспознанным, если для него не существует реконструированного события с положением пика, отличным от исходного менее чем на 10 бинов.
4. Реконструированное событие считается ложным распознаванием, если для него не существует исходного события с положением пика, отличным от реконструированного менее чем на 10 бинов.

5.3.5 Алгоритм 1. Simple

Сперва был испробован тривиальный способ выделения параметров событий - поиск в кадре локальных экстремумов выше порога. Амплитуда и положение события определяется из номера бина экстремума и его значении. Алгоритм крайне быстр в обработке и прост в реализации но страдает низкой точностью - для разделения наложений ему требуется наличие видимого изгиба между близкими событиями. Также, точность полученных амплитуды и положения могут принимать только кратные значению канала и шагу оцифровки соответственно - промежуточных значений быть не может. Стоит отметить, что

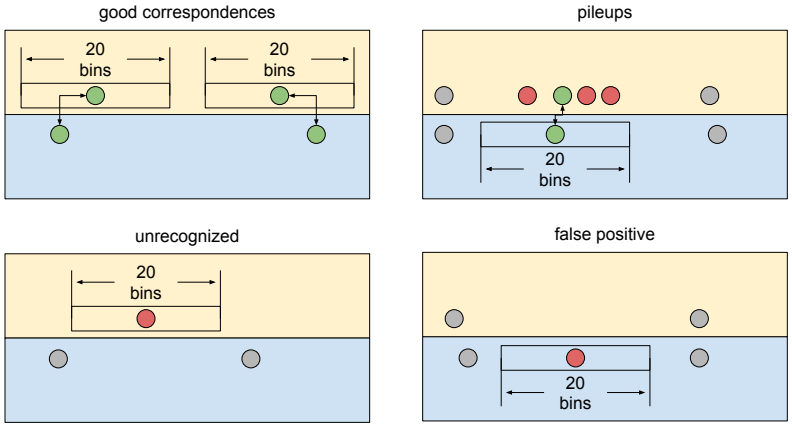

Рисунок 5.9 — Правила определения соответствий между событиями. Обозначения: желтый прямоугольник содержит исходные события, голубой - восстановленные, зелеными кружками отмечены взаимно соответствующие друг другу события, красным - нераспознанные \ложнораспознанные \наложенные события в зависимости от примера, серыми - не относящиеся к примеру события, нарисованные для наглядности.

из-за своей простоты алгоритм по сути не использует форму события вообще и соответственно не проводит связанных с ней коррекций, таких как коррекция на выбросы от предыдущих событий.

Результаты тестирования изображены на рисунке 5.10. На тестовом кадре не распознаны события 2 и 3. Второе событие не имеет перегиба на границе с первым, поэтому не может быть распознано как локальный экстремум и пропускается. Третье событие, хоть и имеет перегиб, попало на выброс от событий 1 и 2 и стало ниже порогового значения. Событие 4 было успешно распознано, однако из-за попадания на выбросы предыдущих событий его амплитуда исказилась. Искажение также можно увидеть на гистограмме амплитудных ошибок - асимметрия спектра справа (амплитуда исходного события больше реконструированного) вызвана отсутствием поправки на выбросы.

Несмотря на качество обработки, алгоритм все еще является быстрым и может быть полезен - на установке «Троицк ню-масс» мы используем его для визуализации набираемых данных в реальном времени как наиболее подходящий для этого.

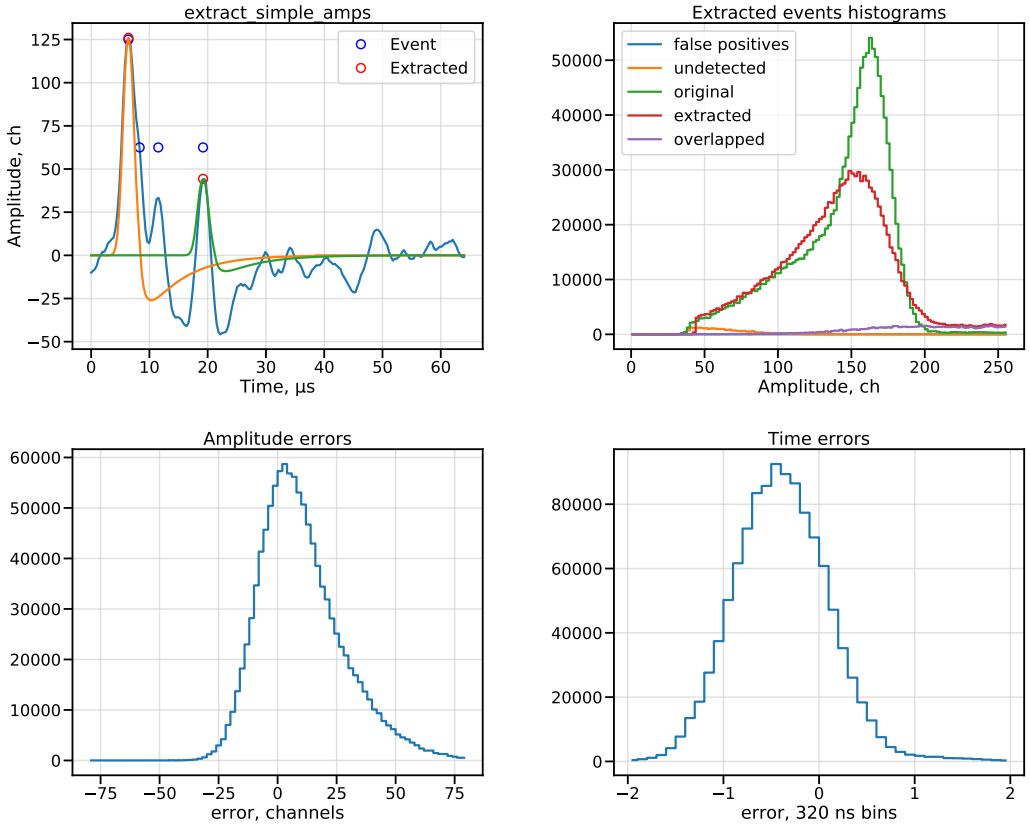

Рисунок 5.10 — Результаты тестирования алгоритма 1. На верхнем левом графике приведен пример обработки на конкретном примере. Справа сверху представлены распределения всех типов событий. Внизу слева и справа показаны гистограммы ошибок по амплитудам и положениям соответственно.

5.3.6 Алгоритм 2. Approximate After-pulses

Алгоритм является улучшенным вариантом Simple с добавлением коррекции на выбросы. Добавлены новые правила:

1. После определения положений событий через локальные экстремумы, считывание амплитуд осуществляется последовательно от события к событию строго слева направо.
2. После считывания амплитуды в бине события, из кадра удаляется форма, соответствующая событию с параметрами текущего извлеченного события. Т.о. из данных удаляется само событие и его влияние на другие события. Т.к. обход идет слева направо, амплитуда обрабатываемого события будет скорректированной, т.к. все более ранние события уже обработаны и удалены из кадра.

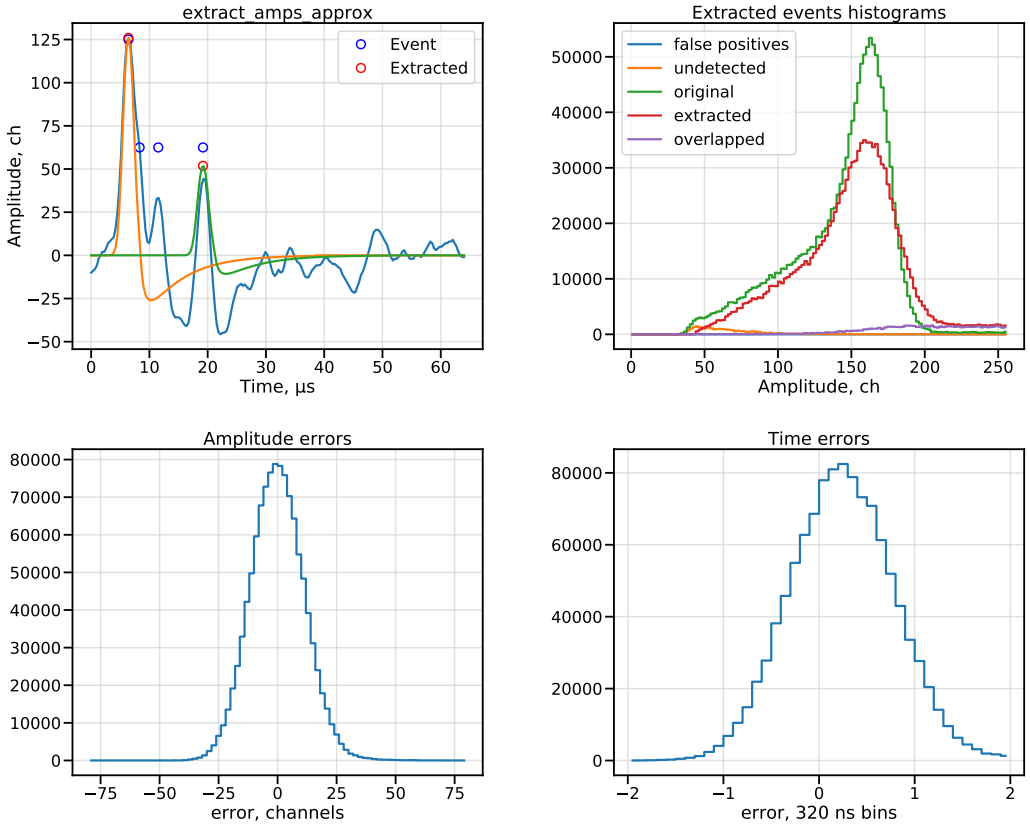

Рисунок 5.11 — Результаты тестирования алгоритма 2

Результаты тестирования показаны на рисунке 5.11.Как и в Simple, положения событий определяются только один раз вначале. Такой выбор был сделан ради скорости и стабильности обработки. Из-за этого, по той же причине что и в Simple события 2 и 3 остались не распознанными. Однако теперь амплитуда четвертого события скорректировалась (неполностью, т.к. события 2 и 3 не были удалены из кадра). Коррекцию на выбросы можно также наблюдать на распределении ошибок восстановления амплитуд - форма стала заметно симметричнее. Потенциально, алгоритм можно ускорить и добиться производительности на уровне Simple. Для этого вместо вычитания всей формы события из кадра нужно вычитать ее только из бинов с пиками событий.

5.3.7 Алгоритм 3. Front Fit

Данный алгоритм разрабатывался с целью уменьшить количество нераспознанных событий с помощью обнаружения новых событий после коррекции на выбросы.

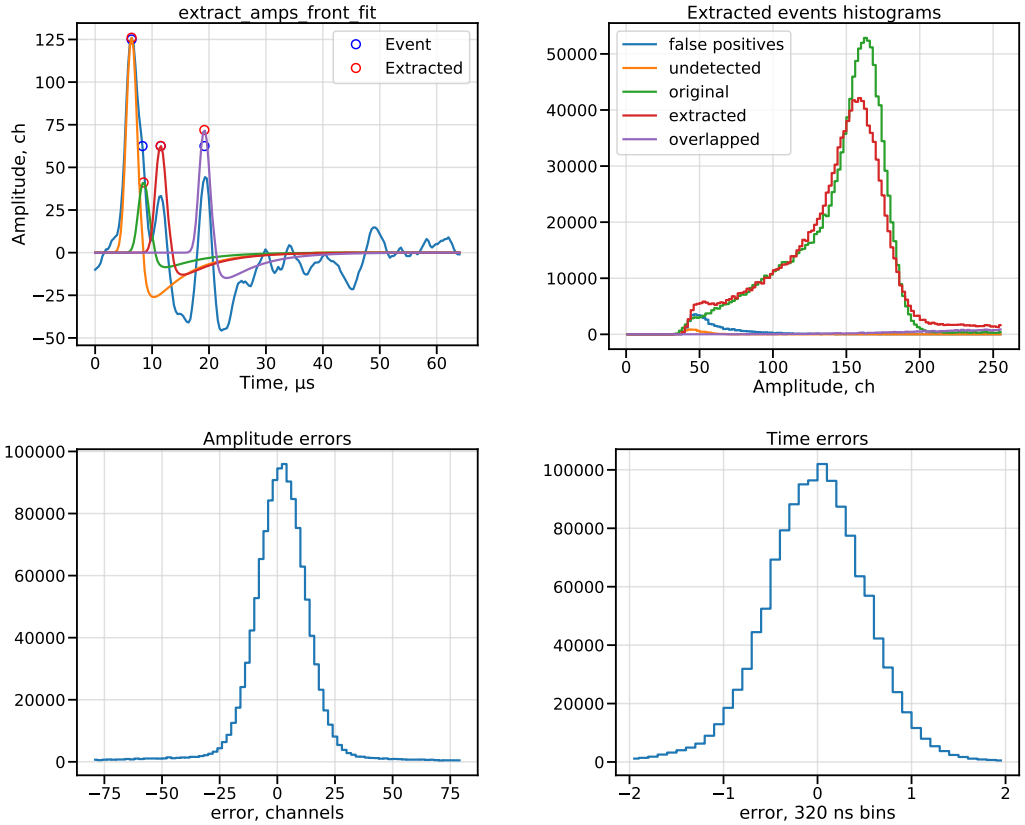

Рисунок 5.12 — Результаты тестирования алгоритма 3

Изменен способ поиска - теперь за итерацию ищется только одно событие (самое раннее). Условие поиска - первое превышение порога в кадре. После нахождения алгоритм производит фитирование формы события по нескольким точкам фронта (окрестность порогового бина) и восстанавливает параметры. Выбор окрестности фитирования обусловлен временной очередностью: самый ранний фронт в кадре чаще всего является фронтом первого события и поэтому не искажается. После извлечения параметров, реконструированная форма удаляется из кадра и процесс повторяется.

Такой подход позволяет:

– Определять наложенные события не имеющие разделения в виде перегиба.
– Определять события, изначально имеющие амплитуду ниже порога.
– За счет фитирования точность извлекаемых параметров меньше размера бина по положению и меньше канал по амплитуде.

Результаты тестирования показаны на рисунке 5.12. Все события в тестовом кадре успешно распознаны. Спектры амплитудных и временных ошибок симметричны. По восстановленной амплитудной гистограмме видно, что алгоритм распознал намного больше событий по сравнению с алгоритмами 1 и

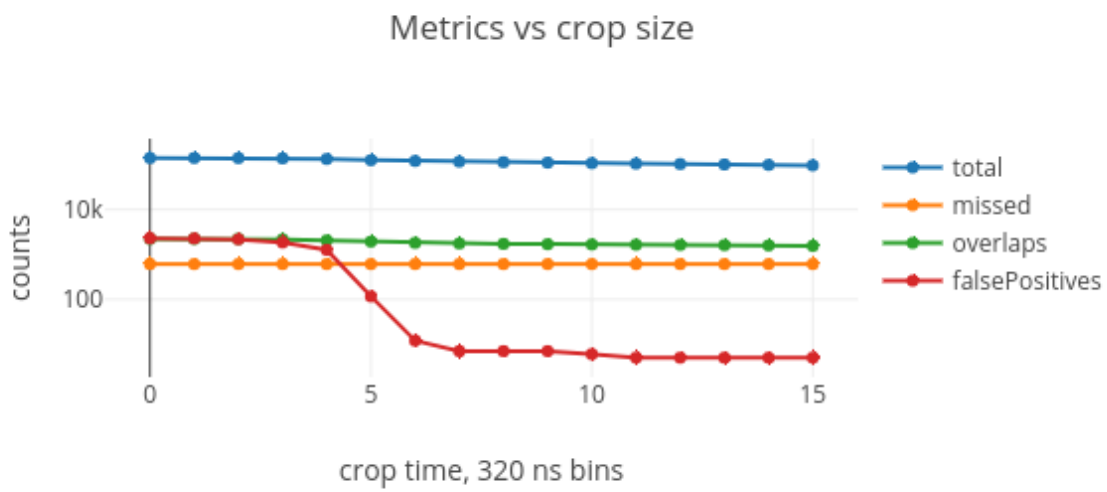

Рисунок 5.13 — Соотношение типов событий в зависимости от размера обрезки.

2. Однако вместе с этим увеличилось количество ложноположительных срабатываний (стало около 2% против 0.01% у алгоритмов 1 и 2). Такой большой процент вносит значительное и неизвестное искажение при обработке и делает бессмысленным дальнейший анализ данных.

Нозиком А.А. был предложен вариант постобработки позволяющий эффективно фильтровать ложные события. Способ заключается в:

1. введению искусственного мертвого времени;
2. вырезание групп восстановленных событий, в которых каждое событие имеет соседнее событие на расстояние меньше чем искусственное мертвое время, вместе с занимаемыми бинами;
3. корректировка времени набора на вырезанные области.

За счет особенностей распределения Пуассона, фильтр удаляет из выборки в основном аномальные группы близких событий, не подчиняющиеся ему. Также удаляется часть настоящих событий, однако зная время обрезки и их подчинение процессу Пуассона, можно скорректировать создаваемое искажение. Более подробно метод описан в [53].

На рисунке 5.13 изображен график зависимости количества событий по типам от вводимого мертвого времени. При мертвом времени равным 6 бинам (1.92 мкс) постобработка удаляет практически все двойные наложения (до 0.01%) и 15% всех событий.

	Simple	Approx spikes	Front Fit (with filter)
Время работы, %	61.1	158.8	9771.5
Распознаваний, %	88.34	88.29	97.999
Нераспознанных, %	11.66	11.71	02.01
Ложноположительные,%	0.01	0.01	0.01
Мертвое время, мкс	3.195	3.167	1.92

Таблица 11 — Сравнение алгоритмов

5.3.8 Сравнение

В таблице 11 приведены результаты тестирования всех трех алгоритмов (время сбора данных составляет 35 с и скорость счета составляет 40 кГц). Эффективное мертвое время определяется формулой 5.3.

Первый и второй алгоритмы дают одинаковые количественные результаты и имеют одинаковое эффективное мертвое время около 3 мкс. Однако второй метод корректирует амплитуды на выбросы от предыдущих событий. Потенциально из-за коррекции алгоритм 2 может извлечь больше событий, но это негативно скажется на его стабильности и производительности. Алгоритм 3 с фильтрацией по искусственному мертвому времени имеет мертвое время 2 мкс (равно введенному искусственному), выделяет практически на 10% больше событий и имеет точность выше единиц АЦП. Помимо достоинств, алгоритм «Front Fit» имеет некоторые недостатки, такие как:

- Использование фильтрации уменьшает выборку на 15% и соответственно уменьшает статистическую точность результатов.
- Алгоритм работает очень медленно, т.к. написан на Python и в обработке производит фитирование кривой.

5.4 Обобщение алгоритмов и оптимизация системы обработки

Продолжительная эксплуатация реализованных алгоритмов выделения параметров событий показала следующие недостатки:

1. Скорость. За несколько сеансов с Лан10-12PCI объем набранных данных достиг 500 Гб и вопрос производительности встал на первое место.
2. Появилось требование к точности времени реконструированного события. Точное время необходимо для проведения соответствующего временного анализа и процедур контроля качества, описанных в [53]. Алгоритмы 1, 2 используют простые методы поиска пиков, которые дают время только с точностью в один бин, чего недостаточно. Алгоритм 3 обладает достаточной точностью, однако скорость работы не позволяет серьезно использовать его.
3. Ограниченность инструментария. Невозможность использования функций при обработке. Необходимость реализации алгоритмов в виде тензорных преобразований.
4. Обобщение алгоритмов. При анализе данных было обнаружено, что форма события меняется, притом новая форма не имеет аналитической формулы. Также оказалось, что свойства шума тоже меняются.

Для увеличения производительности, было решено сменить язык программирования с Python на Kotlin([54]), который статически компилируется, имеет преимущество совместимости с библиотеками Java, имеет современный и удобный синтаксис и возможность использовать библиотеки на нескольких платформах компиляции в будущем. Переход на Kotlin позволил увеличить скорость анализа до десяти раз по сравнению с версией Python с той же функциональностью и использовать недоступные на Python возможности.

5.4.1 Обобщенный генератор шума

Для сеанса 2017_11 был построен амплитуд спектр шума. В нем обнаружились расхождения со ранее используемом в [15] спектром. В связи с этим был также перестроен спектр для 2017_05. Как оказалось, он также не совпадает со спектром из статьи и больше похож на шум сеанса 2017_11. Скорее всего это связано с увеличением скорости счета и появлением базовой линии в сигнале.

Использование старой реализации шумогенератора на датасете 2017_11 потребовало бы повторной ручной подстройки количества этапов и степеней сглаживания под новые данные. С учетом перевода кода на Kotlin такую под-

стройку стало затруднительно проводить. Поэтому мы решили изменить способ генерации шума, обобщив его на произвольный вид шума.

Работа обобщенного генератора шума разделена на 2 части:

1. Создание индексированной выборки фрагментов шумов
2. Генерация шума по созданной выборке

Для создания выборки фрагментов шумов используются также начала в набранных кадрах (в нашем генераторе мы берем первые 35 бинов). Элемент выборки - фрагмент из начала кадра максимальной длины (и не меньше заданной - 10 бинов), пересекающий нуль с обоих концов. В качестве индекса берется пара из значений первых двух бинов. Выборка записывается на диск в виде двух файлов - бинарный файл с последовательно записанными фрагментами и файл с сериализованной хеш картой, содержащей списки отступов фрагментов в файле для каждого индекса (алгоритм был реализован в нескольких вариантах, в т.ч. с использованием СУБД в качестве хранилища фрагментов, однако вариант с двумя файлами оказался самым быстрым).

Генерация шума происходит следующим образом:

1. Из бинарного файла считывается произвольный фрагмент
2. При запросе из начала текущего фрагмента извлекается отдается один бин. Операция повторяется пока длина текущего запроса больше 2 бинов.
3. Оставшиеся 2 бина текущего фрагмента преобразовываются в индекс, по которому из хеш карты берется список отступов фрагментов имеющих этот индекс.
4. Из списка случайным образом выбирается отступ, по нему из файла считывается фрагмент
5. Считанный фрагмент становится текущим. Переход к п.2.

На рисунке 5.14 показан принцип работы генератора.

Таким образом мы генерируем шум, склеенный из реальных фрагментов. Совпадение 2 бинов на границе примыкающих фрагментов обеспечивает гладкость перехода. Т.к. для генерации используются только реальные данные - статистически, генерируемый шум будет неотличим от настоящего, что подтверждается сравнением, показанном на рисунке 5.15. Это также означает, что алгоритму не требуется дополнительных ручных настроек под конкретный тип шума.

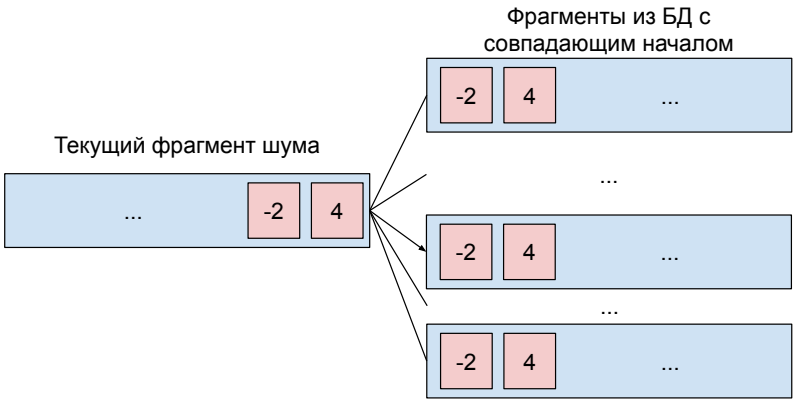

Рисунок 5.14 — Принцип работы обобщенного генератора шума.

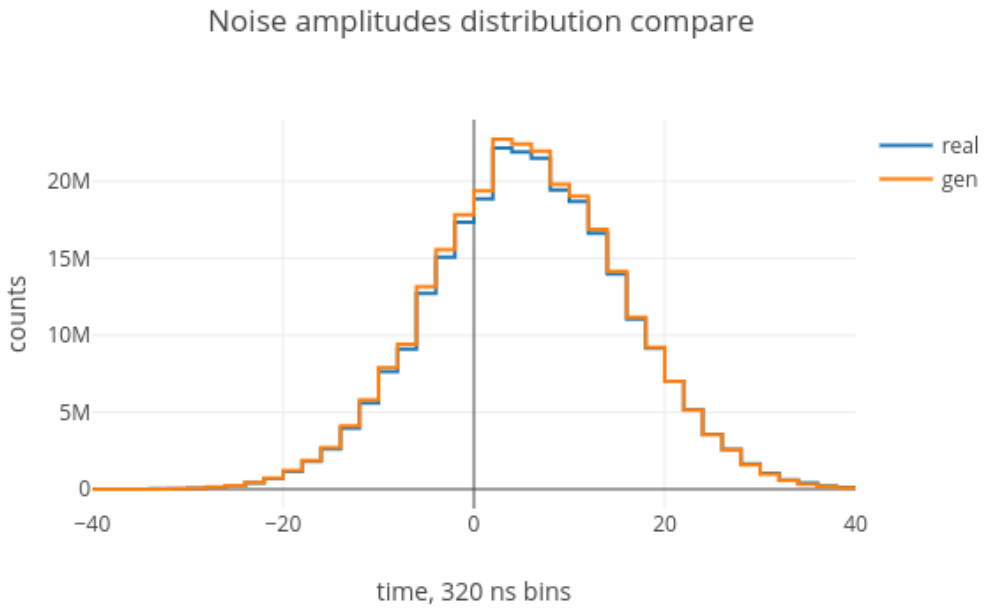

Рисунок 5.15 — Сравнение шумовых спектров.

5.4.2 Обобщенный генератор формы события

Анализ данных показал, что формы событий в датасете 2017_11 также отличаются от сеанса 2017_05 (см рисунок 5.16). Более того, теперь в выбросе после события появились осцилляции и подобрать аналитическую функцию для описания формы уже не получится. Всвязи с этим мы решили обобщить алгоритм генерации события и убрать из него ручную настройку.

Также как и в старом генераторе, описанном в , сначала строятся усредненные формы событий по группам амплитуд. Затем, по формам строятся интерполяционные сплайны, по которым находятся точные положения пиков.

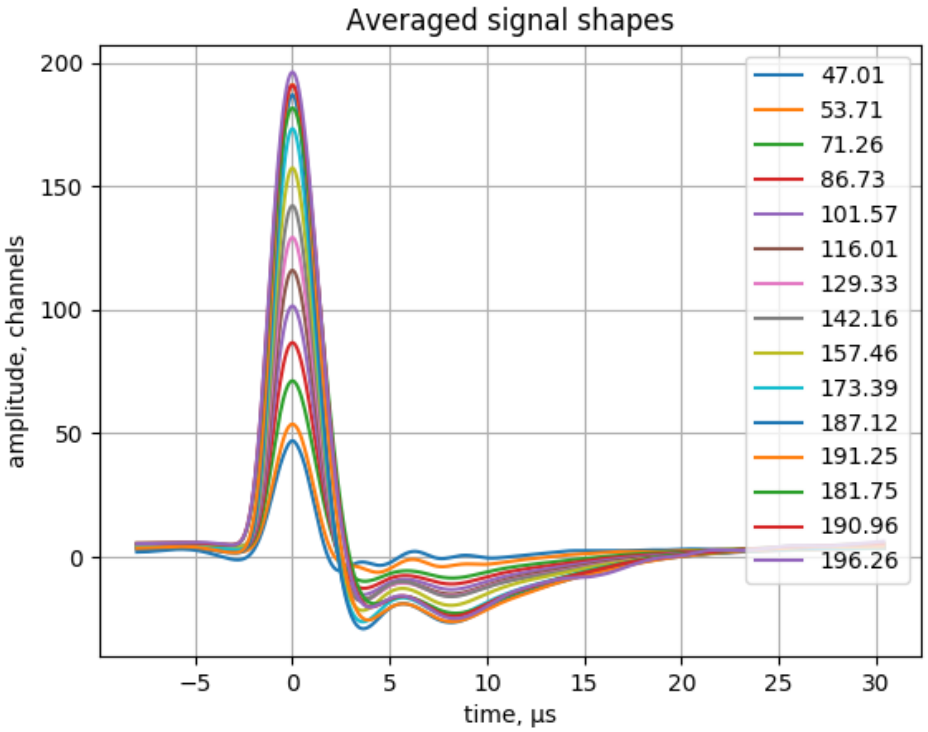

Рисунок 5.16 — Усредненные формы событий.

Отцентрированные по пику сплайны форм переносятся на новые временные сетки, имеющие одинаковые координаты узлов. Из набора сеток для амплитудных групп строится двумерная сетка, амплитудными координатами узлов является амплитуда пика усредненной формы в группе, временными координатами - координаты узлов формы (они одинаковые для всех групп). На полученной сетке строится бикубический сплайн.

Если параметры точки генерируемого события находятся внутри области интерполяции (в нашем генераторе область интерполяции слегка усекается по амплитудному диапазону, т.к. на границах интерполятор выдает артефакты) - значение определяется интерполяцией. Если амплитуда генерируемой точки выходит за пределы области интерполяции - значение определяется линейной экстраполяцией граничного значения. Если положение генерируемой точки выходит за пределы области интерполяции - значение приравнивается нулю вне зависимости от ее амплитуды.

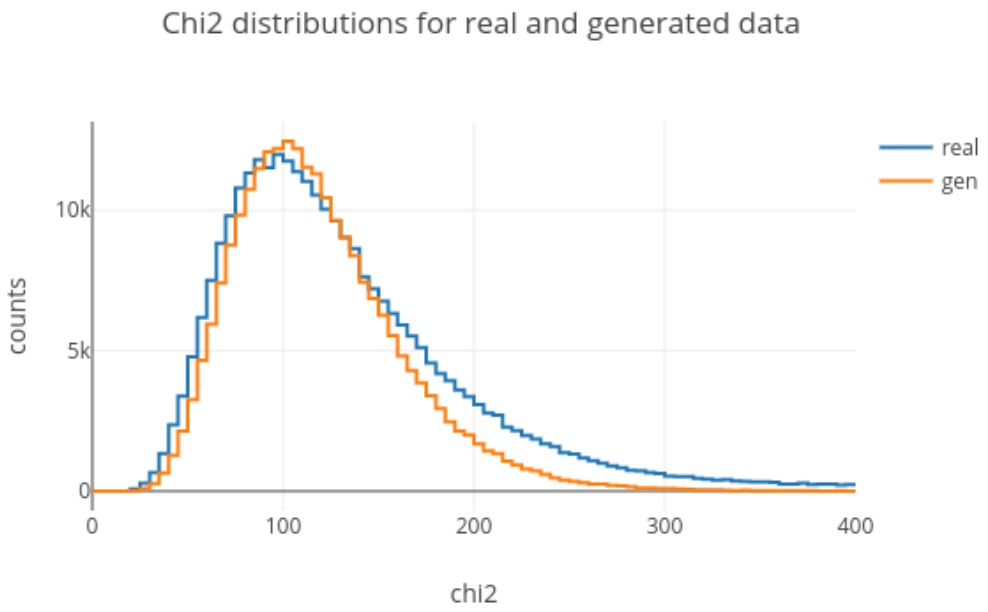

Рисунок 5.17 — Спектр отклонений χ^2.

5.4.3 Валидация генератора

Аналогично подглаве 5.3.3 были построены χ^2 отклонения для реального и смоделированного сигналов. Параметры событий извлекались параболической интерполяцией по 5 точкам в окрестности локального максимума выше порога. Результат показан на рисунке 5.17. Среднее значение для обоих распределений одинаково, χ^2 для реального датасета выглядит более размытым. По проведенному сравнению χ^2 распределений можно сказать, что алгоритм моделирования создает совпадающий с реальным сигнал.

5.4.4 Алгоритм 4. Fast Fit

Этот алгоритм написан на Kotlin. В основе лежит Алгоритм 3. Некоторые этапы изменены в целях ускорения работы. Схему работы можно описать следующей последовательностью:

	Время работы, %	распозн., %	нераспозн., %	ложнопол. ,%	мертвое время, мкс
Fast Fit	405.7	96.57	3.45	0.03	2.24

Таблица 12 — Метрики алгоритма 4

1. Поиск самого левого локального максимума выше порога (максимум определяется как самый большой бин в плавающем окне размером 5 бинов)
2. Извлечение параметров события с помощью параболического фитирования по 5 точкам в окрестности локального максимума.
3. Сохранение полученных параметров.
4. Вычитание из кадра реконструированной по параметрам формы события.
5. Переход к п.1.

По сравнению с Алгоритмом 3, здесь изменен способ определения параметров события - используется параболическое фитирование в окрестности пика. Плюсы такого подхода - ускорение за счет упрощенного фитирования и более устойчивая область фитирования (т.к. отношение сигнала к шуму выше, чем на фронте события). Недостатком является более высокий шанс пропустить самое раннее событие и попасть на следующее, имеющее искажение.

Результаты тестирования представлены на рисунке 5.18 и в таблице 12. По результатам тестирования алгоритмы 3 и 4 похожи. Fast Fit генерирует похожее число двойных наложений. Их также можно убрать с помощью фильтрации в обмен на 15% выборки. Преимуществом алгоритма является скорость - превышающая алгоритм 3 в 24 раза.

Для подробного анализа в проект были добавлены программы, которые выводят графики обработки кадра в случаях, если в нем произошло наложение, появилось ложноположительное срабатывание или потеряно событие.

Просмотрев несколько кадров , мы заметили что наложения практически всегда имеют одну и ту же природу: расстояние и отношения амплитуд между исходными сигналами появляются в такой комбинации, что суммарная форма по ширине близка к ширине одиночного события с пиком амплитуды примерно равным сумме исходных амплитуд.

В случае, когда исходные события находятся чуть дальше друг от друга и суммарная форма превышает ширину одиночного события, алгоритм может

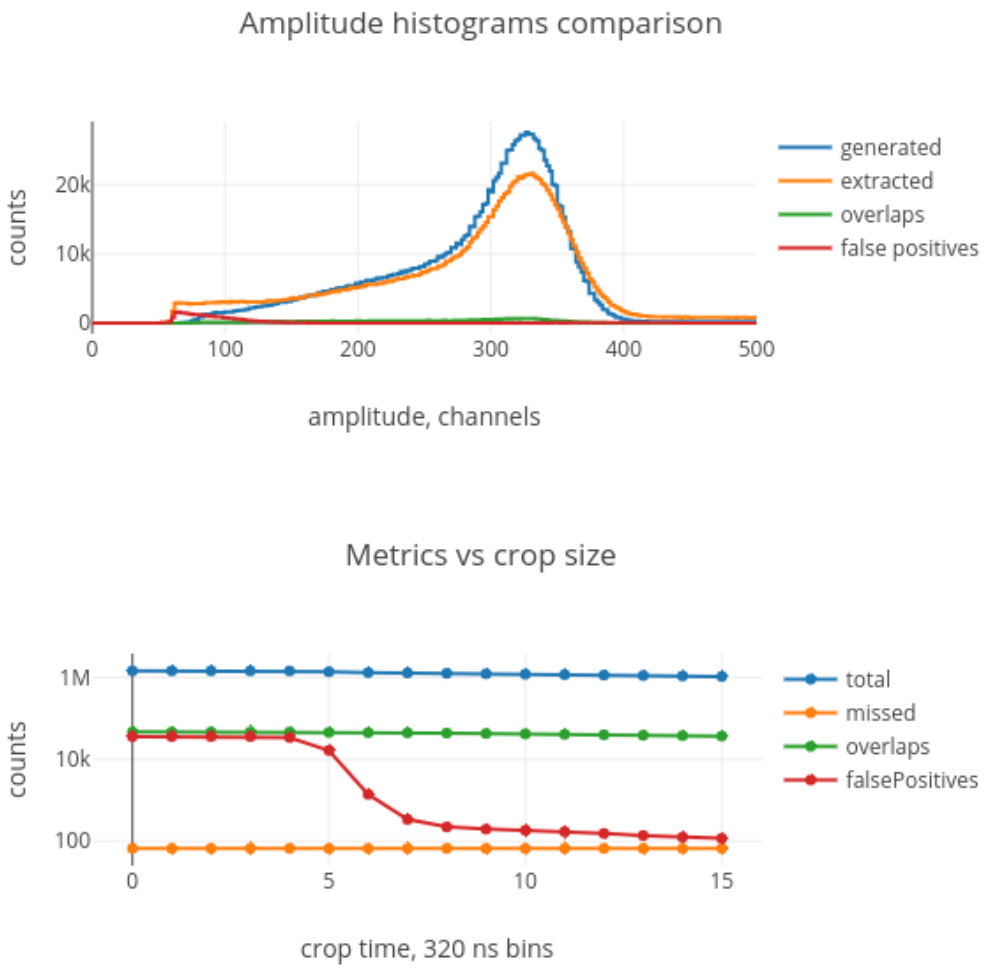

Рисунок 5.18 — Результаты тестирования алгоритма 4

генерировать ложноположительное срабатывание. Просмотр таких случаев показал, что это происходит по двум сценариям:

1. Два события расположены относительно симметрично и пик их суммарной формы находится посередине исходных пиков. В этом случае алгоритм привяжется к пику конечной формы (из-за параболического фитирования в области максимума) и выделит несуществующее событие. Реальные исходные события как правило тоже выделяются, но с меньшими амплитудами, т.к. часть их ушла на ложноположительное. Пример такого случая изображен в таблице ниже справа.
2. Два близких события образуют несимметричную конченную форму. В этом случае, как правило, выделяются оба исходных события выделяются достаточно хорошо, однако из-за ошибок определения пиков (связанных со способом поиска) после выделения исходных событий в форме остается «излишек», который интерпретируется как событие.

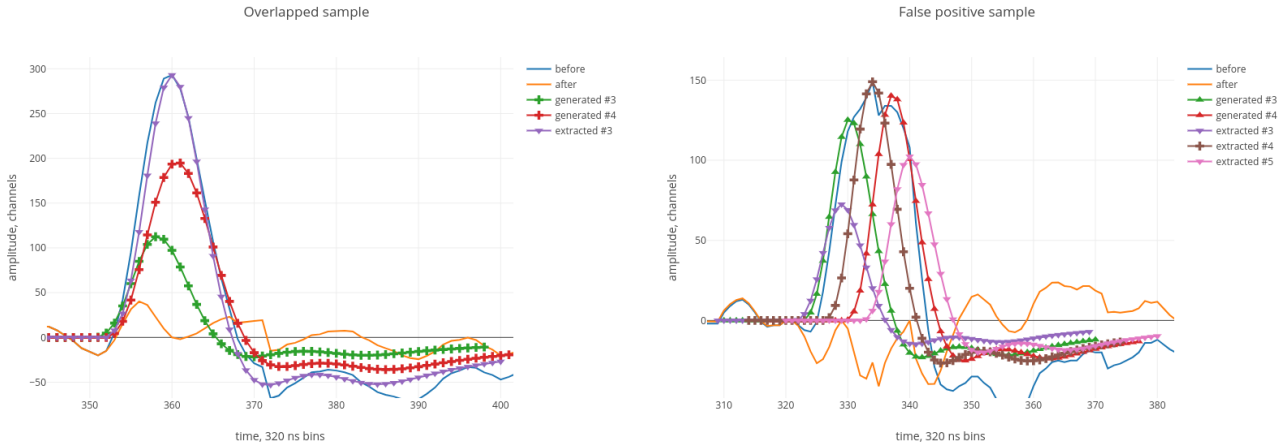

Рисунок 5.19 — Типичные случаи наложения (слева) и ложного срабатывания (справа).

Обычно ложноположительное срабатывание такого типа имеет небольшую амплитуду и может быть эффективно отфильтровано.

Оба сценария проиллюстрированы на рисунке 5.19.

5.4.5 Алгоритм 5. Double Fit

Во процессе просмотра случаев выделения параметров событий, при которых возникали ложноположительные срабатывания была выявлена закономерность - в большинстве таких ситуаций происходит также нарушение очередности обработки наложенных событий, т.е. следующее выделенное событие будет иметь более раннее время возникновения чем текущее. Т.о. нарушение очередности можно использовать как сигнал наличия наложения.

Алгоритм Double Fit представляет собой алгоритм Fast Fit с дополнительной обработкой в случае нарушения очередности.

Схема работы:

1. Запись текущего состояния в стек.
2. Выделение события аналогично алгоритму Fast Fit.
3. Проверка очередности последних двух событий.
    а) Переход к п.1, если очередность соблюдена или расстояние между событиями больше 8 бинов.
    б) в противном случае:

	Время работы, %	распозн., %	нераспозн., %	ложнопол. ,%	мертвое время, мкс
Double Fit	591.4	96.3	3.7	0.03	0.96

Таблица 13 — Метрики алгоритма 5

1) Откат состояния до этапа, на котором выделенное событие было левее текущего. Одновременно ищутся самое левое и самое правое события отмененные откатом.
2) Выделение двойного наложения фитированием. Функция - сумма двух форм событий, область интерполяции - промежуток между крайними отмененными событиями.
3) Запись извлеченной пары событий.
4) Вычитание извлеченных событий из кадра.
5) Переход к п.1.

Блок-схема алгоритма изображена на рисунке 5.20. Обработка опирается на стек состояний, хранящий историю предыдущих этапов и возможность отката на предыдущие состояния. При откате назад выделенные на текущем этапе события удаляются из массива результатов, а кадр обращает вычитание формы из себя.

Т.е. при нарушении очередности алгоритм возвращает состояния до последнего правильного и меняет способ выделения на фитирование одновременно двух событий по полной форме.

В результате качество выделения значительно улучшается - алгоритм генерирует 0.03% ложноположительных срабатываний и не требует дополнительных постобработок. Результаты тестирования приведены на рисунке 5.21 и в таблице 13.

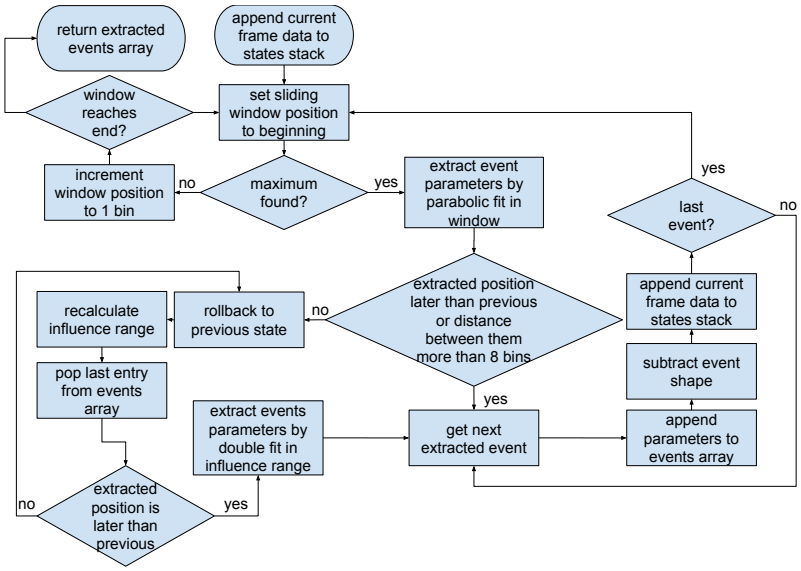

Рисунок 5.20 — Блок-схема алгоритма «Double Fit».

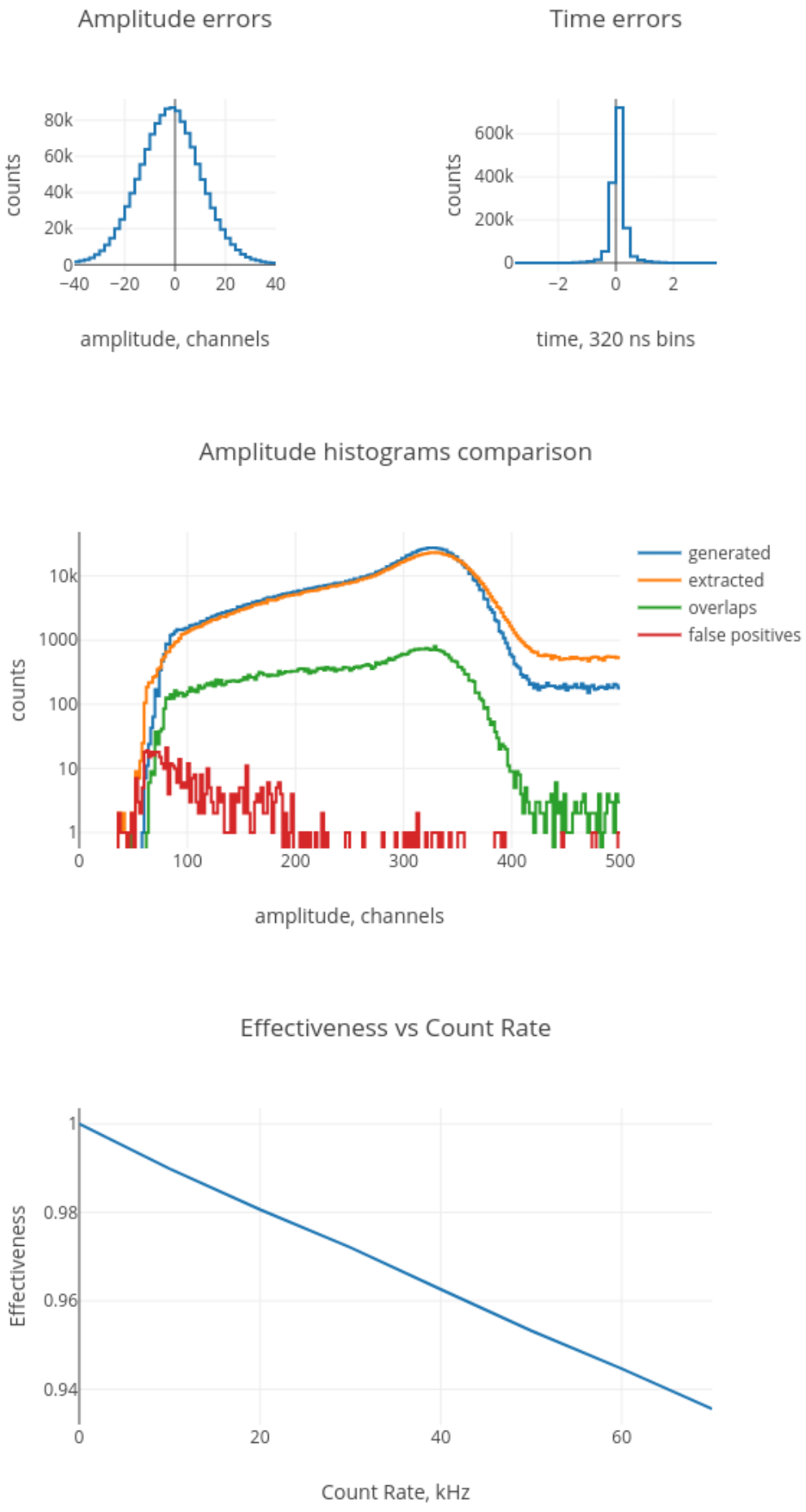

Рисунок 5.21 — Результаты тестирования алгоритма 5

Заключение

Основные результаты работы заключаются в следующем.

1. Проведена общая модернизация системы сбора данных установки «Троицк ню-масс». Разработана модульная архитектура, позволяющая проводить изолированную разработку и отладку отдельных подсистем. Исходный алгоритм управления переписан на C++/Qt и перенесен на новую архитектуру. Разработаны инструменты визуального контроля качества набора в реальном времени.

   Разработан и реализован оригинальный формат передачи и хранения данных DataForge Envelope, оптимизированный под задачи экспериментальной физики[9]. Написаны реализации протокола на C++(Qt), Python, Java. Преимущества формата:

   – Единица данных - пакет, содержащий метаданные и бинарные данные. Метаданные имеют текстовый формат и используются для хранения параметров набора и настроек. Объединение метаданных и бинарных данных в один пакет исключает потерю части набранных данных при манипуляциях с ними.
   – DataForge Envelope может использоваться как для хранения, так и для передачи данных. В качестве хранилища используется файловое дерево, каждый пакет записывается в виде файла, таким образом для просмотра данных требуется только текстовый редактор. По служебным параметрам пакета можно определить его размер, что позволяет создавать вложенные сообщения и использовать формат для работы в потоке стека TCP/IP.
   – Возможно быстрое чтение и предобработка сохраненных данных только по метаданным.

   Реализована распределенная система сбора данных. Алгоритм управления установкой разбит на три независимых модуля, отвечающих за: считывание сигнала с детектора, управление напряжением спектрометра и проведение набора по сценарию. Модули имеют единообразный программный интерфейс и взаимодействуют через стек TCP/IP. В основе аппаратной модернизации лежит использование ПК-контроллера

крейтов КАМАК CCPC7 для управления подсистемами установки. Каждый модуль представляет собой программный сервер, который устанавливается в CCPC7 и осуществляет контроль аппаратуры через магистраль крейта, RS-232, RS-485 интерфейсы и другие специфические соединения. Для упрощения разработки реализована работа модулей в виртуальном режиме, в котором не требуется подключение к реальной аппаратной части. Виртуальный режим упрощает разработку и позволяет проводить неполный запуск системы (отключенные модули заменяются виртуальными). Проведено успешное тестирование, в результате которого подтверждено полное соответствие функционалу исходной системы.

2. Проведен переход на запись непрерывного сигнала. Для модернизированной системы сбора разработан модуль записи данных непрерывной оцифровки с помощью АЦП Лан10-12PCI. В процессе записи плата последовательно сбрасывает кадры максимальной длины по программному триггеру. Каждый последующий кадр сбрасывается сразу после сохранения предыдущего. Таким образом, сохраняется непрерывная оцифровка сигнала с пропусками на время сбросов. Такой подход позволяет обойти проблему аппаратного мертвого времени, возникающую при стандартном наборе по триггеру, и проводить более сложную обработку сигнала в офлайн-режиме. Особенности модуля Лан10-12PCI:
   - Модуль полностью совместим с программным интерфейсом системы и не требует модификации других модулей при интеграции/отключении.
   - Модуль прозрачно встраивается в систему (т. е. подключается между управляющим и детекторным модулями, транслируя их сообщения без изменений) и набирает данные в дублирующем режиме.
   - Для взаимодействия с АЦП Лан10-12PCI используется написанное в рамках работы [14] CLI-приложение.
3. В рамках сотрудничества с TRISTAN на установке «Троицк ню-масс» проведено подключение прототипа детектора производства XGLab для эксперимента KATRIN к системе сбора данных установки. Разработан эмулятор сигналов, работающий по спецификации, предоставленной

XGLab. При помощи эмулятора создан программный пакет, обеспечивающий взаимодействие с АЦП DANTE, управление считыванием, а также хранение событий вместе со специфическими для этой системы метаданными. При помощи этой системы проведены два сеанса измерений на установке с использованием детектора TRISTAN.

4. Разработаны алгоритмы выделения параметров событий (амплитуды и положения) из оцифрованного сигнала, снятого платой Лан10-12PCI. Для данных «Троицк ню-масс» при выделении параметров событий удалось добиться рекордного эффективного мертвого времени порядка 0.9 мкс для средней длины «колокола» события 6 мкс и размера одного канала оцифровки 320 нс (мертвое время при аппаратной обработке событий составляло около 7 мкс). Также обработка формы импульса позволила исключить систематическую ошибку восстановления амплитуд, вызванную наложением на хвосты предыдущих событий. Проведено обобщение алгоритмов на произвольную форму импульса. Производительность алгоритма оптимизирована до уровня, когда он может быть использован в режиме реального времени. Исходный код с инструкциями по воспроизведению результатов выложен в открытый репозиторий[55][56]. Описание алгоритмов опубликовано в [15].
   - Разработаны обобщенные генераторы шума и событий, симулирующие реальный сигнал АЦП. Генераторы извлекают параметры из кадров АЦП и не требуют ручной настройки. Разработаны тесты для оценки качества генерируемых данных.
   - Разработан фреймворк для тестирования алгоритмов обработки, имеющий функционал: расчета процентного соотношения событий по типам (нераспознанные, ложнораспознанные и т. д.), построения распределения ошибок распознавания амплитуды и времени, графического отображения случайного примера кадра, вызывающего ошибку обработки.
   - Разработан алгоритм выделения параметров событий, имеющий мертвое время 0.9 мкс. В основе его работы лежит последовательное вычитание из кадра распознанных событий от первого к последнему. Для выделения параметров используются два метода: параболическое фитирование пика для одиночных или хорошо разделенных событий и фитирование

функцией суммы форм двух событий для наложенных событий. Условием для использования второго метода выступает нарушение очередности извлекаемого события.

Проделанная работа представляет ценность как в рамках эксперимента «Троицк ню-масс», так и для развития экспериментальной физики частиц в целом.

Распределенные системы контроля и сбора данных представляют существенный интерес для экспериментов следующего поколения. Успешный опыт использования этой концепции в эксперименте «Троицк ню-масс» будет применен при разработке новых систем. Распределенная система хранения данных и обмена сообщениями также представляет интерес как для экспериментальной физики, так и для промышленного использования.

Существенной особенностью разработанных алгоритмов по разделению наложений является то, что они не привязаны к конкретной форме сигнала и могут быть в перспективе адаптированы для использования в других экспериментах.

Все программные компоненты, которые реализованы в данной работе и имеют потенциал повторного использования, были подробно задокументированы и выложены в репозитории с открытым доступом.

В заключение автор выражает благодарность и большую признательность научному руководителю Нозику А. А. за поддержку, помощь, обсуждение результатов и научное руководство. Также автор благодарит Пантуева В.С. за помощь во время подготовки материала и написания диссертации, Абдурашитова Д. Н., Берлева А. И. и Задорожного С.В. за помощь в оформлении материалов, технические консультации и активное участие в разработке системы сбора данных.

Список литературы

1. *W. Walter, C.* The Super-Kamiokande Experiment / C. W. Walter. — 2008. — Mar.
2. The Sudbury Neutrino Observatory / A. Bellerive [et al.] // Nuclear Physics B. — 2016. — Apr. — Vol. 908.
3. First measurements in search for keV sterile neutrino in tritium beta-decay in the Troitsk nu-mass experiment / J. N. Abdurashitov [et al.] // JETP Letters. — 2017. — June. — Vol. 105, no. 12. — P. 753—757.
4. A White Paper on keV Sterile Neutrino Dark Matter / R. Adhikari [et al.] // ArXiv e-prints. — 2016. — Feb. — Vol. 2017.
5. The current status of "Troitsk nu-mass" experiment in search for sterile neutrino / D. N. Abdurashitov [et al.] // Journal of Instrumentation. — 2015. — Apr. — Vol. 10, no. 10. — T10005—T10005.
6. Microservices: Yesterday, Today, and Tomorrow / N. Dragoni [et al.] // Present and Ulterior Software Engineering. — Springer International Publishing, 2017. — P. 195—216.
7. *Al-Debagy, O.* A Comparative Review of Microservices and Monolithic Architectures / O. Al-Debagy, P. Martinek. — 2019. — eprint: arXiv:1905.07997.
8. AFI Electronics: CCPC7. — URL: https://afi.jinr.ru/CCPC7.
9. DataForge Streaming envelope format documentation. — 2019. — URL: http://npm.mipt.ru/dataforge/docs.html#envelopes.
10. A novel detector system for KATRIN to search for keV-scale sterile neutrinos / S. Mertens [et al.] // Journal of Physics G: Nuclear and Particle Physics. — 2019. — May. — Vol. 46, no. 6. — P. 065203.
11. *KATRIN Collaboration.* KATRIN design report 2004 : tech. rep. / KATRIN Collaboration, KATRIN Collaboration ; Forschungszentrum, Karlsruhe. — 2005. — 245 p. — 51.54.01; LK 01.

12. Measurements with a TRISTAN prototype detector system at the "Troitsk nu-mass" experiment in integral and differential mode / T. Brunst [et al.] // Journal of Instrumentation. — 2019. — Nov. — Vol. 14, no. 11. — P11013—P11013.

13. Silicon drift detector prototypes for the keV-scale sterile neutrino search with TRISTAN / K. Altenmüller [et al.] // Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. — 2018. — Dec. — Vol. 912. — P. 333—337.

14. *Abdurashitov, D. N.* The Long-Term Stability of a Fused-Silica Proportional Counter / D. N. Abdurashitov, V. G. Chernov // Instruments and Experimental Techniques. — 2019. — Jan. — Vol. 62. — P. 5—9.

15. *Chernov, V.* Shape-based event pileup separation in Troitsk nu-mass experiment / V. Chernov, A. Nozik // Journal of Instrumentation. — 2019. — Aug. — Vol. 14. — T08001—T08001.

16. *Hill, J.* An Alternative Analysis of the LSND Neutrino Oscillation Search Data on $\nu^{-}\mu \to \nu^{-}e$ / J. Hill // Physical Review Letters - PHYS REV LETT. — 1995. — Oct. — Vol. 75. — P. 2654—2657.

17. Light Sterile Neutrinos: A White Paper / K. N. Abazajian [et al.]. — 2012. — Apr.

18. *Smoot, G.* See Saw Inflation / Dark Energy / G. Smoot. — 2014. — May.

19. *Shrock, R.* New tests for and bounds on neutrino masses and lepton mixing / R. Shrock // Physics Letters B. — 1980. — Oct. — Vol. 96. — P. 159—164.

20. Role of Sterile Neutrino Warm Dark Matter in Rhenium and Tritium Beta Decays / H. J. de Vega [et al.] // Nuclear Physics B. — 2011. — Sept. — Vol. 866.

21. *Rodejohann, W.* Signatures of Extra Dimensional Sterile Neutrinos / W. Rodejohann, H. Zhang // Physics Letters B. — 2014. — July. — Vol. 737.

22. *Barry, J.* Sterile neutrinos and right-handed currents in KATRIN / J. Barry, J. Heeck, W. Rodejohann // Journal of High Energy Physics. — 2014. — Apr. — Vol. 2014.

23. Current Direct Neutrino Mass Experiments / G. Drexlin [et al.] // Advances in High Energy Physics. — 2012. — Oct. — Vol. 2013.

24. *Monreal, B.* Relativistic Cyclotron Radiation Detection of Tritium Decay Electrons as a New Technique for Measuring the Neutrino Mass / B. Monreal, J. Formaggio // Physical Review D. — 2009. — Apr. — Vol. 80.

25. Single-Electron Detection and Spectroscopy via Relativistic Cyclotron Radiation / D. M. Asner [et al.] // Physical review letters. — 2014. — Aug. — Vol. 114.

26. Development of a Relic Neutrino Detection Experiment at PTOLEMY: Princeton Tritium Observatory for Light, Early-Universe, Massive-Neutrino Yield / S. Betts [et al.]. — 2013. — July.

27. Direct Measurement of the Mass Difference of ^{163}Ho and ^{163}Dy Solves the Q-Value Puzzle for the Neutrino Mass Determination / S. Eliseev [et al.] // Physical Review Letters. — 2015. — Aug. — Vol. 115. — P. 062501.

28. *Kopp, J.* Ultra-low Q values for neutrino mass measurements / J. Kopp, A. Merle // Physical Review C. — 2009. — Nov. — Vol. 81.

29. The electron capture 163Ho experiment ECHo / K. Blaum [et al.] // Journal of Low Temperature Physics. — 2014. — May. — Vol. 176. — P. 876—884.

30. HOLMES / B. Alpert [et al.] // The European Physical Journal C. — 2015. — Apr. — Vol. 75, no. 3.

31. NuMecs Position Paper. — URL: http://p25ext.lanl.gov/%E2%88%BCkunde/NuMECS/.

32. *Liao, W.* keV scale ν_R dark matter and its detection in β decay experiments / W. Liao // Phys. Rev. D. — 2010. — Oct. — Vol. 82.

33. *Li, Y.* Possible capture of keV sterile neutrino dark matter on radioactive β-decaying nuclei / Y. Li, Z.-z. Xing // Physics Letters B. — 2011. — Jan. — Vol. 695, no. 1—4. — P. 205—210.

34. Криогенная система для эксперимента по измерению массы покоя электронного антинейтрино : preprint / А. И. Белесев [и др.] ; ИЯИ АН СССР. — Москва, 1988. — П—0614.

35. Сверхпроводящая система спектрометра для измерения массы покоя электронного антинейтрино : preprint / А. И. Белесев [и др.] ; ИЯИ АН СССР. — Москва. — П—0615.

36. *С.В., З.* Поиск массы нейтрино в бета-распаде трития. Система сбора данных и первичная обработка результатов. : дис. ... канд. / С.В. Задорожный. — Москва : ИЯИ РАН, 2004.

37. *Urban, E.* Parser for multipart/form-data / E. Urban. — 2013. — URL: https://github.com/defnull/multipart/commit/1e435e0c99bfa70b27c16b3e6fd9f4c87098ca

38. DataForge. — 2015. — URL: http://npm.mipt.ru/dataforge/.

39. *Wikipedia.* Shebang (Unix) — Wikipedia, The Free Encyclopedia / Wikipedia. — 2019. — [Online; accessed 03-June-2019]. http://en.wikipedia.org/w/index.php?title=Shebang%20(Unix)&oldid=898134985.

40. Protocol Buffers. — 2019. — URL: https://developers.google.com/protocol-buffers/.

41. *Deutsch, P.* ZLIB Compressed Data Format Specification version 3.3 / P. Deutsch, J.-L. Gailly. — Internet Engineering Task Force, 05/1996. — URL: http://www.ietf.org/rfc/rfc1950.txt. RFC 1950 (Informational).

42. python-df-parser. — 2018. — URL: https://github.com/kapot65/python-df-parser.

43. counter-redirecter. — 2017. — URL: https://github.com/kapot65/counter-redirecter.

44. python-df-tcp. — 2017. — URL: https://github.com/kapot65/python-df-tcp.

45. lan10-viewer. — 2017. — URL: https://github.com/kapot65/lan10-viewer.

46. Detector Development for a Sterile Neutrino Search with the KATRIN Experiment / T. Brunst [et al.]. — 2018. — Jan.

47. dante-server. — 2017. — URL: https://bitbucket.org/Kapot/dante-server.

48. AutoIt Scripting Language - AutoIt. — URL: https://www.autoitscript.com/site/autoit/.

49. МЕТОД ИЗМЕРЕНИЯ БОЛЬШИХ СКОРОСТЕЙ СЧЕТА ДЕТЕКТОРОВ ИОНИЗИРУЮЩЕГО ИЗЛУЧЕНИЯ С ВЫСОКОЙ ТОЧНОСТЬЮ / Д. Н. АБДУРАШИТОВ [и др.] // ПРИБОРЫ И ТЕХНИКА ЭКСПЕРИМЕНТА. — 2006. — Июль. — Т. 2. — 169b—169.

50. Pileup Mitigation with Machine Learning (PUMML) / P. T. Komiske [et al.] // Journal of High Energy Physics. — 2017. — July. — Vol. 2017.

51. Analytical modeling of pulse-pileup distortion using the true pulse shape; applications to Fermi-GBM / V. Chaplin [et al.] // Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. — 2012. — Nov. — Vol. 717.

52. ЛАн10-12PCI/ЛАн10-12PCI-У. — 2017. — URL: http://www.rudshel.ru/show.php?dev=13.

53. *Nozik, A.* Statistical time analysis for regular events with high count rate / A. Nozik // Journal of Instrumentation. — 2019. — June. — Vol. 14. — P06008—P06008.

54. *Jemerov, D.* Kotlin in Action / D. Jemerov, S. Isakova. — Manning Publications Company, 2016.

55. signal-utils. — 2019. — URL: https://github.com/kapot65/signal-utils.

56. lan10-processing. — 2019. — URL: https://bitbucket.org/Kapot/lan10-processing.

Список рисунков

Список таблиц