

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Новосибирский национальный исследовательский государственный университет»
(Новосибирский государственный университет, НГУ)
Структурное подразделение Новосибирского государственного университета –
Высший колледж информатики Университета (ВКИ НГУ)
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Направление подготовки: 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

Образовательная программа: 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

**РАЗРАБОТКА АДАПТИВНОЙ СИСТЕМЫ
АВТОМАТИЗАЦИИ УСКОРИТЕЛЬНОГО ИСТОЧНИКА
ЭПИТЕПЛОВЫХ НЕЙТРОНОВ БНЗТ ИЯФ СО РАН**

утверждена приказом по ВКИ НГУ № 02/3-204 от «27» апреля 2017 г.

Кошкарева Алексея Михайловича, группа 14214 _____
(фамилия, имя, отчество студента) (подпись студента)

«К защите допущена»

Заведующий кафедрой, к.ф.-м.н
(ученая степень, звание)

Попов Л.К. / _____
(ФИО) / (подпись)

«___» _____ 20__г.

Руководитель ВКР

д-р физ.-мат. наук, в.н.с, зав. лаб. БНЗТ НГУ
(ученая степень, звание)

Таскаев С.Ю., / _____
(ФИО) / (подпись)

«___» _____ 20__г.

Дата защиты: «___» _____ 20__г.

Новосибирск
2017

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ	4
ВВЕДЕНИЕ.....	5
1 ПОСТАНОВКА ЗАДАЧИ	6
1.1 Описание предметной области	6
1.2 Описание ускорителя.....	6
1.3 Формулировка задачи.....	7
1.4 Описание автоматизированных устройств.....	8
1.4.1 Охлаждаемые диафрагмы	8
1.4.1.1 Принцип работы.....	8
1.4.1.2 Измерение по трем точкам	10
1.4.1.3 Звуковое оповещение	11
1.4.2 Пирометр.....	12
1.4.3 Шиберы	14
1.4.4 Пищалка 300В	14
1.4.5 Измеритель мощности на Li мишени	15
1.4.6 Ионный источник.....	17
1.4.7 Ребутатор	18
1.4.8 Тепловизор Flir T650SC	19
1.4.9 Пути развития системы	21
1.5 Функциональные требования к системе управления	21
2 АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧИ.....	22
2.1 Начальная конфигурация ускорителя.....	22
2.2 Предлагаемая конфигурация	24
2.3 Конфигурация программы	26
2.4 Ведение журнала	26
2.4.1 Общий принцип	26
2.4.2 Ведение журнала в устаревшей программе	27
2.4.3 Перехват ошибок в программе	27
3 АРХИТЕКТУРА ПРОГРАММЫ УПРАВЛЕНИЯ.....	28
3.1 Устройства	29
3.2 Каналы.....	29

3.2.1 Реальные каналы	32
3.2.2 Виртуальные каналы	32
3.3 Модули	32
3.4 Утилиты	33
3.5 Разработанная программа управления.....	33
3.5.1 Форма «Основное»	34
3.5.2 Форма «Управление».....	35
3.5.3 Форма «Флюенс»	36
3.5.4 Форма «Источник Н».....	36
3.5.5 Форма «Ребутатор».....	37
3.5.6 Форма «Все каналы»	38
4 РАЗРАБОТКА БИБЛИОТЕК.....	40
4.1 ModuleWizard.....	40
4.1.1 Общий принцип	40
4.1.2 Пример	42
4.2 LogWizard.....	42
4.2.1 FormLog.....	42
4.2.2 TimeLog.....	43
4.3 GraphWizard	43
4.3.1 Компонент «Векторный радар».....	44
4.3.2 Компонент «Шкала значения»	45
4.3.3 Компонент «Лампочка»	46
4.3.4 Компонент «Стрелочный индикатор»	46
4.3.5 Компонент «График реального времени»	47
5 ОТЛАДКА И ТЕСТИРОВАНИЕ.....	49
ЗАКЛЮЧЕНИЕ	50
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	51
ПРИЛОЖЕНИЕ А	52
ПРИЛОЖЕНИЕ Б.....	54
ПРИЛОЖЕНИЕ В	64
ПРИЛОЖЕНИЕ Г	67

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

БНЗТ	Бор-нейтронозахватная терапия.
Зиверт	Единица измерения эффективной и эквивалентной дозы ионизирующего излучения в Международной системе СИ
Система управления	Совокупность модулей сбора данных, программы управления и базы данных.
Децентрализованная система управления	Это система управления, у которой модули сбора данных территориально разнесены и расположены в непосредственной близости с объектом измерения.
Периферия	Опрашиваемое или управляемое устройство ввода / вывода.
АЦП	Аналого-цифровой преобразователь.
ЦАП	Цифро-аналоговый преобразователь.
ЦВх	Цифровой вход.
ЦВых	Цифровой выход.
Супрессор вторичной эмиссии	Устройство для подавления нежелательных электронов.
Флюенс	Интеграл по времени от плотности потока частиц.

ВВЕДЕНИЕ

Целью данной работы является реализация автоматизированной системы управления, контролирующей оборудование экспериментальной установки. В ИЯФ СО РАН осуществляется разработка ускорительного источника нейтронов [1, 2], предназначенного для проведения бор-нейтронозахватной терапии (БНЗТ) [3, 4] злокачественных опухолей в условиях онкологической клиники. Данный метод терапии очень эффективен в отношении ряда неизлечимых в настоящее время радиорезистентных опухолей, например, таких, как глиобластома мозга и метастазы меланомы.

Все существующие исследования БНЗТ успешно проведены при помощи ядерных реакторов, но из-за их нестабильности требуется создание безопасного ускорителя. Именно такой ускоритель разрабатывается в Институте ядерной физики, и для его дальнейшего развития требуется создание новой системы автоматизации на базе промышленного оборудования.

Спецификой ускорителя является его постоянная модернизация и внедрение диагностик разного типа. Для стабильной работы ускорителя необходимо создание гибкой и масштабируемой системы автоматизации, позволяющей управлять подготовкой ускорителя к работе, осуществлять проведение экспериментов и выключать ускоритель после работы с минимальным участием оператора.

Отсутствие гибкой, масштабируемой и централизованной системы управления сдерживает внедрение ускорителя в клиническую практику. Решение этой проблемы описывается в данной работе.

1 ПОСТАНОВКА ЗАДАЧИ

1.1 Описание предметной области

Ускоритель состоит из множества устройств, таких как: вакуумные насосы, вакуумные шиберы, источники питания, измерительное оборудование и прочие элементы. Для получения на ускорителе оптимальных параметров, позволяющих проводить бор-нейтронозахватную терапию, требуется децентрализованная, масштабируемая и надежная система управления (далее система), обеспечивающая контроль и сбор данных над всеми узлами ускорителя.

Текущая конфигурация ускорителя не позволяет быстро добавить опрашиваемое / управляемое устройство (далее периферия), что неприемлемо в условиях постоянной модификации ускорителя. Поэтому требуется разработка и внедрение новой системы управления.

1.2 Описание ускорителя

Ускоритель расположен на территории ИЯФ СО РАН в специальном радиационно-защищенном бункере. Его схематичное описание, габариты и фотография представлены на рисунке 1 и рисунке 2.

Обслуживание ускорителя затрудняется тем, что в процессе эксперимента радиационный фон может достигать единицы зиверт. Это означает, что за 10 минут человек получит примерно 200 годовых доз. Кроме того, в процессе работы некоторые узлы находятся под потенциалом 1 000 000 вольт, поэтому все манипуляции должны происходить удаленно.

Ускоритель генерирует эпитепловые нейтроны. Для сравнения, тепловыми нейтронами называют нейтроны с энергиями до 0,5 эВ, быстрыми – более 10 кэВ, а вот в промежутке от 0,5 эВ до 10 кэВ – эпитепловые. Для БНЗТ нужны эпитепловые, поскольку тепловые глубоко не проникают, быстрые проникают глубоко, но дают ненужную дозу за счет упругого рассеяния, а вот эпитепловые и проникают туда куда надо и импульс у них уже маленький, чтобы давать дозу за счет рассеяния.

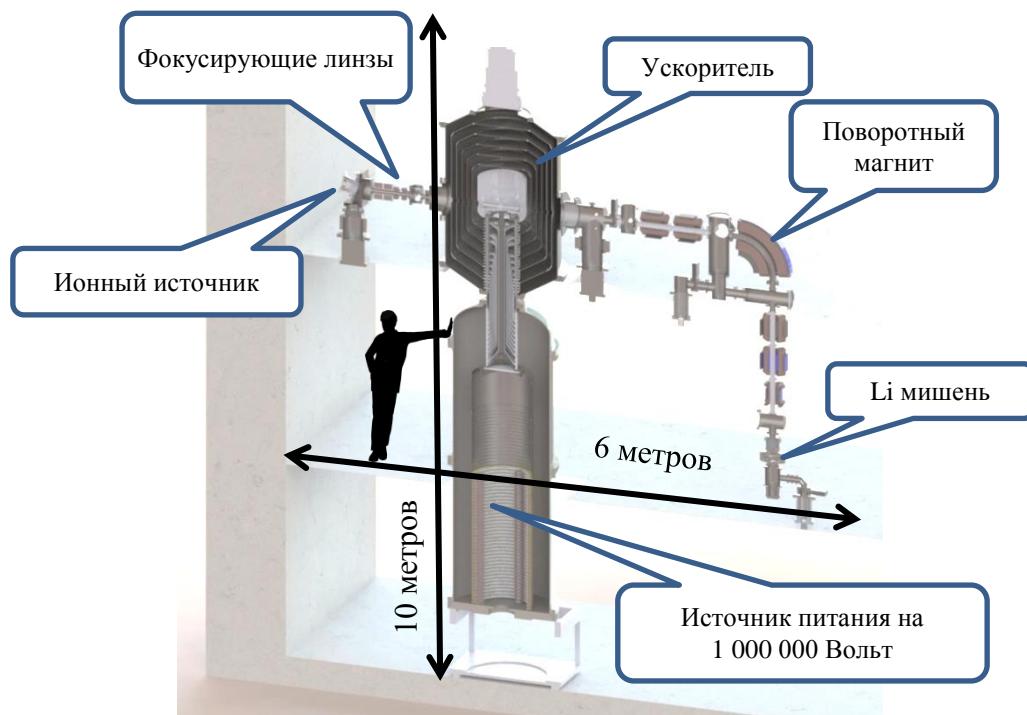


Рисунок 1 – Схема экспериментального ускорителя

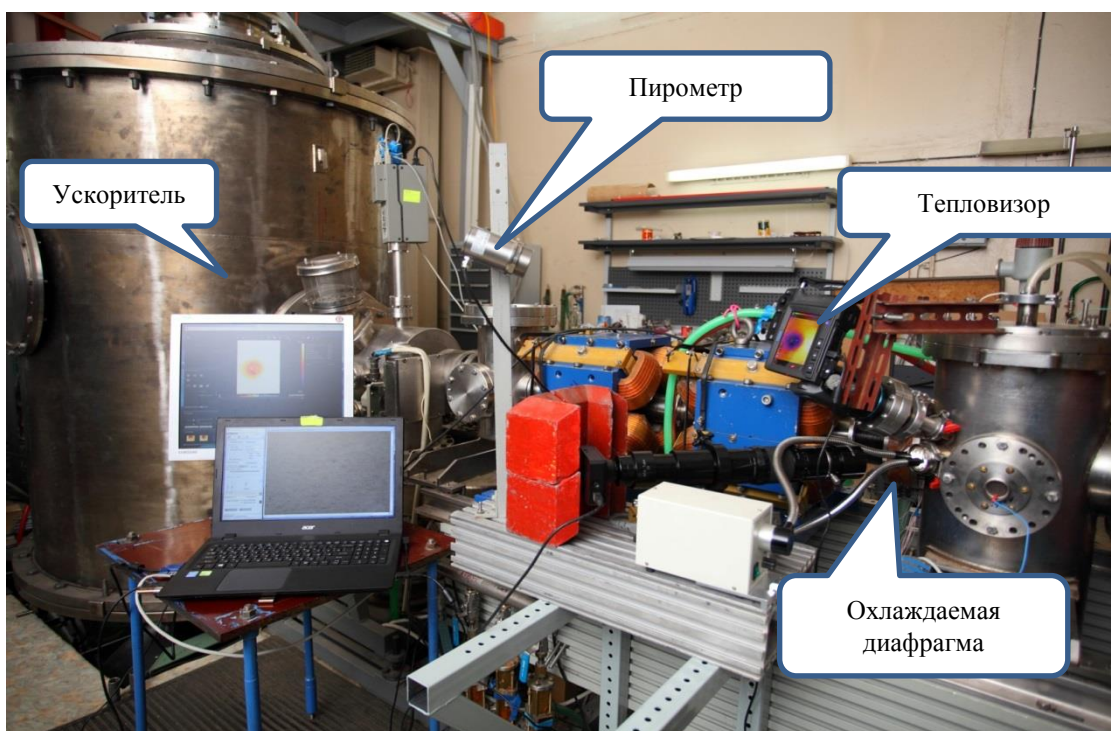


Рисунок 2 – Фото экспериментального ускорителя

1.3 Формулировка задачи

Целью работы являлась реализация системы автоматизации ускорителя БНЗТ, а именно: разработка программы оператора, сопряжение периферии с

программой управления, отображение данных оператору и ведение журнала эксперимента. На рисунке 3 указаны автоматизированные устройства.

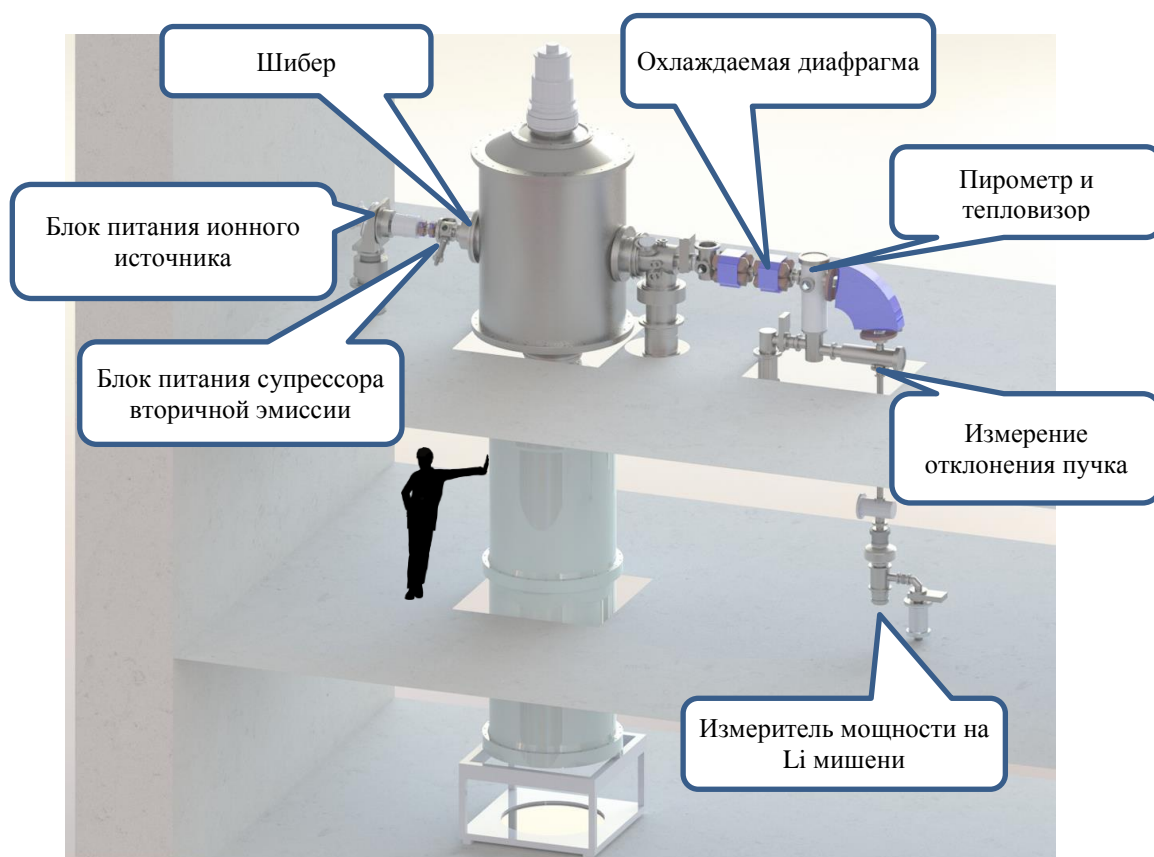


Рисунок 3 – Автоматизированные устройства

1.4 Описание автоматизированных устройств

1.4.1 Охлаждаемые диафрагмы

1.4.1.1 Принцип работы

Охлаждаемые диафрагмы – это своего рода ограничители, которые отсекают пучок диаметром более 30 мм. Пучок, который высадился на диафрагму, нагревает ее, а этот нагрев регистрируется четырьмя термосопротивлениями. По значениям с термодатчиков можно определить отклонение пучка (подробнее далее). На данный момент установлена только одна диафрагма, но в дальнейшем они будут стоять по всему тракту пучка. Фотографии диафрагм приведены на рисунке 4 и рисунке 5.

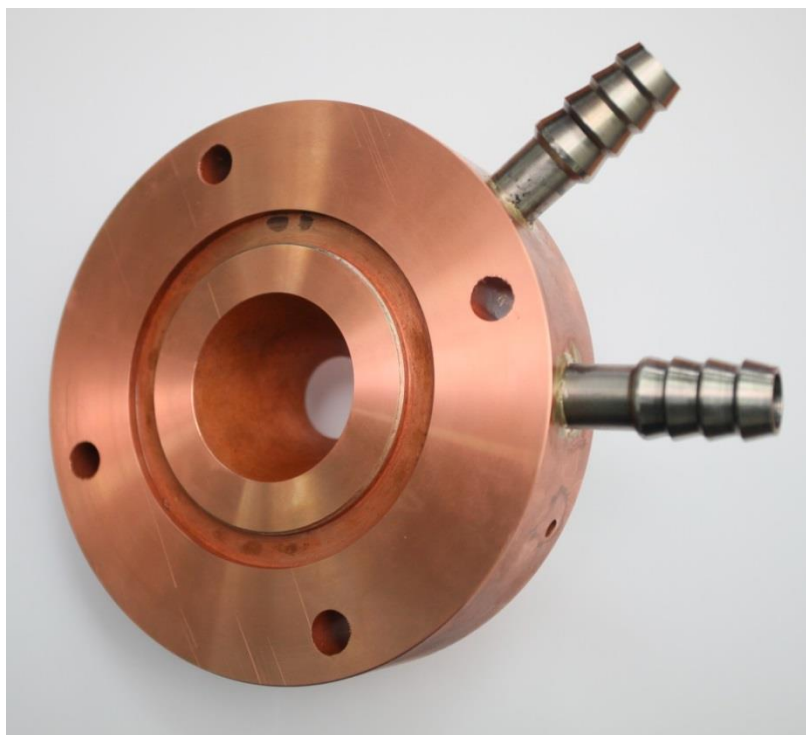


Рисунок 4 – Фотография охлаждаемой диафрагмы

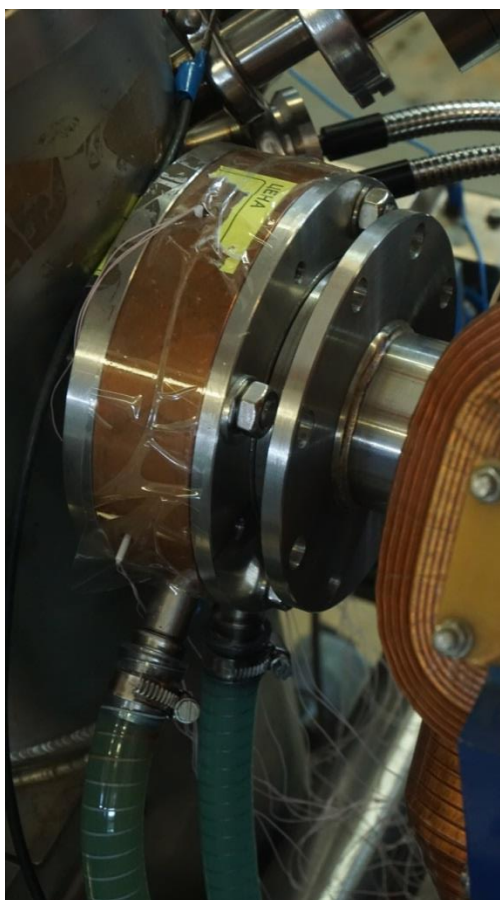


Рисунок 5 – Фотография установленной охлаждаемой диафрагмы

В диафрагме установлено четыре термосопротивления Pt100 для измерения отклонения пучка. Два датчика расположены на горизонтали, и два по вертикали. Вектор отклонения пучка определяется по следующим формулам:

$$d_x = Right - Left, \quad d_y = Up - Down,$$

$$len = \sqrt{d_x^2 + d_y^2},$$

$$angleRad = Atan2(dy, dx),$$

$$angleDeg = -\left(\frac{angleRad * 180}{\pi} - 90\right).$$

Таким образом, зная отклонение в градусах и длину вектора, отклонение можно отобразить оператору, что и было реализовано в данной работе. Панель отклонения пучка и схематическое расположение датчиков показано на рисунке 6.



Рисунок 6 – Панель отклонения пучка: а) значение с охлаждаемой диафрагмы; б) значения после поворотного магнита

1.4.1.2 Измерение по трем точкам

Для проведения эксперимента по генерации нейтронов с литиевой мишени потребовалось диагностировать отклонение пучка после поворотного магнита. Для этого на тракт пучка было установлено три термодатчика, так как

на момент установки диагностики был только один модуль измерения температуры с семь каналами. Четыре уже были заняты охлаждаемой диафрагмой, оставалось только три. Схематическое расположение датчиков показано на рисунке 6. Формулы расчета отклонения представлены ниже:

$$x_A = \text{Cos}(\text{DecToRad}(-90)) * A, \quad y_A = \text{Sin}(\text{DecToRad}(-90)) * A,$$

$$x_B = \text{Cos}(\text{DecToRad}(150)) * B, \quad y_B = \text{Sin}(\text{DecToRad}(150)) * B,$$

$$x_C = \text{Cos}(\text{DecToRad}(30)) * C, \quad y_C = \text{Sin}(\text{DecToRad}(30)) * C,$$

$$\text{final}_x = x_A + x_B + x_C, \quad \text{final}_y = y_A + y_B + y_C,$$

$$\text{len} = \sqrt{\text{final}_x^2 + \text{final}_y^2},$$

$$\text{angleRad} = \text{Atan2}(\text{final}_y, \text{final}_x),$$

$$\text{angleDeg} = -\left(\frac{\text{angleRad} * 180}{\pi} - 90\right).$$

1.4.1.3 Звуковое оповещение

В некоторых ситуациях критично, когда пучок отклоняется от оси тракта. Для оповещения оператора был разработан алгоритм, который оповещает оператора об отклонении.

В первых версиях в качестве оповещения загоралась лампочка. Потом были взяты голосовые оповещения из google переводчика. В последней версии были использованы голосовые команды, начитанные нашей коллегой Лилией (анг. Lilia) из University of science and technology Houari Boumediene, Algeria, которая проходила в нашей лаборатории аспирантуру. Интересно то, что она говорит по-русски *с акцентом*. Именно этот фактор должен привлечь внимание оператора.



Рисунок 7 – Фотография Лилии

1.4.2 Пирометр

Пирометр – устройство дистанционного измерения температуры. Это устройство было использовано для эксперимента, проводимого совместно с Японскими коллегами для изучения эффекта блистеринга в материале при длительном облучении протонным пучком.

При выборе устройства был выбран пирометр Optris 3ML SF CB3 с диапазоном температур от 50 до 400 С° и интерфейсом связи Ethernet. В системе он представляется как виртуальный COM порт. Общение происходит путем приема / передачи определенных команд в виртуальный COM порт. Фотография устройства представлена на рисунке 8.

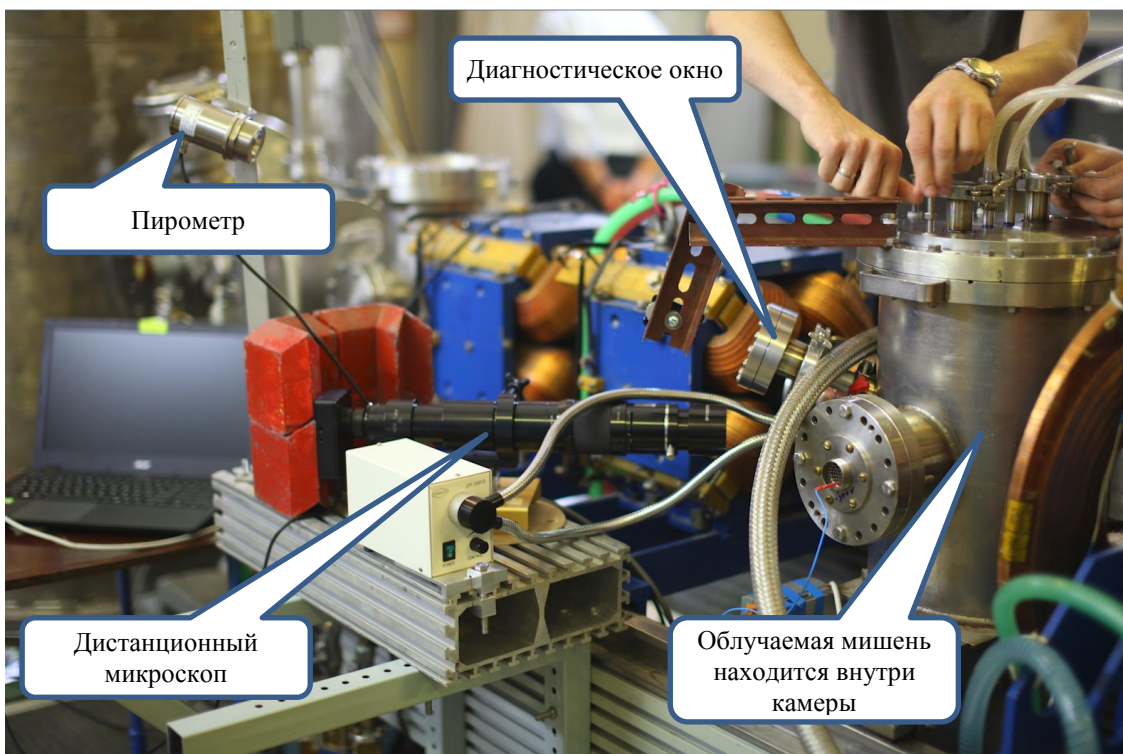


Рисунок 8 – Пирометр на ускорителе

В процессе работы с данным устройством возникли трудности измерения реальной температуры из-за изменения коэффициента вторичной эмиссии. В процессе проведения эксперимента менялась отражающая способность материала, и температура, была не такой, как в реальности. Для этого была проведена калибровка по термопаре.

Показания пирометра отображаются на круглом индикаторе и на графике реального времени. Панели пирометра представлены на рисунке 9.

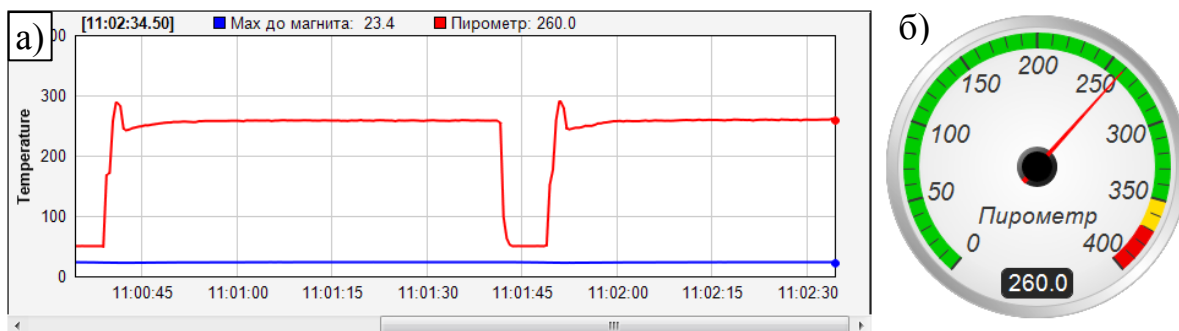


Рисунок 9 – Панель пирометра: а) график реального времени; б) стрелочный индикатор

1.4.3 Шиберы

Шиберы – своего рода заслонки в вакуумных системах. Они требуются для того, чтобы изолировать вакуумные объемы. Для одного эксперимента потребовалось дистанционно открывать и закрывать шибер из пультовой. Для этого были использованы шиберы с выводами, которые нужно замкнуть для открытия, и разомкнуть для закрытия. В качестве модуля периферии был выбран промышленный модуль ADAM 6066 с интерфейсом связи Ethernet и протоколом Modbus. Он имеет шесть каналов реле и шесть каналов цифровых входов (ЦВх). Для данного эксперимента использовался один канал реле. Фото шибера представлены на рисунке 10. Панель управления шибером изображена на рисунке 11.

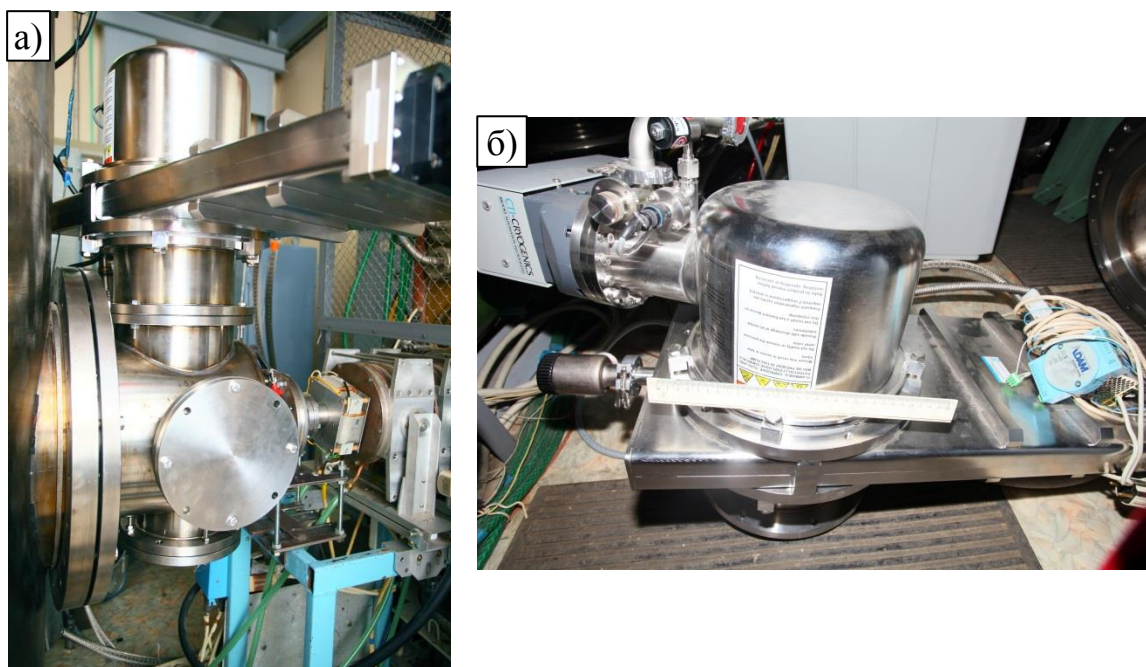


Рисунок 10 – Фото шибера: а) установленный; б) разобранный

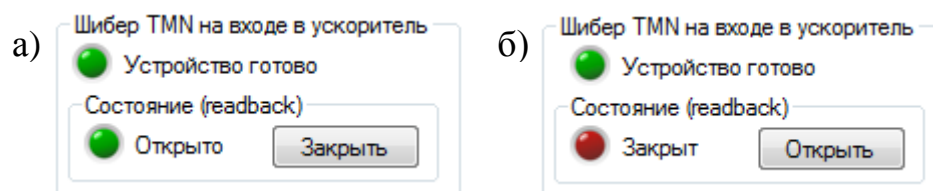


Рисунок 11 – Панель управления шибером: а) открыт; б) закрыт

1.4.4 Пищалка 300В

Для лучшего проведения пучка через тракт ускорителя требуется активировать / деактивировать блок питания супрессора (подавителя) вторичной эмиссии. Для решения этой задачи был использован блок питания на

300 вольт. В первое время он стоял на пультовой, и во время эксперимента его включали и выключали. Из-за характерного писка его прозвали «Пищалкой». Источник питания изображен на рисунке 12.



Рисунок 12 – Источник питания «Пищалка»

Впоследствии его перенесли в бункер, но возможность включения и отключения должна была сохраниться. Для этого был использован тот же модуль, что и в п. 1.4.3. Питание источника проходит через реле АДАМа. Но для эксперимента требовалась обратная связь по включению. Для этого на выходе источника через делитель был подключен цифровой вход (ЦВх) АДАМа.

Таким образом, оператор получил возможность удаленно контролировать состояние блока питания. Панель управления источником питания изображена на рисунке 13.

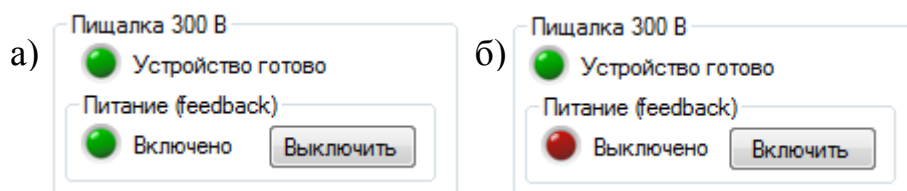


Рисунок 13 – Панель управления источником питания: а) включен; б) выключен

1.4.5 Измеритель мощности на Li мишени

Измеритель мощности на Li мишени – это устройство, которое измеряет мощность, высаживаемую на Li мишень пучком протонов. Расчет мощности происходит на основании измерения входной и выходной температуры воды, а

так же по потоку воды. После измерения, параметры рассчитываются по формуле, и оператору выводится только рассчитанная мощность. Принципиальная схема измерителя представлена на рисунке 14.

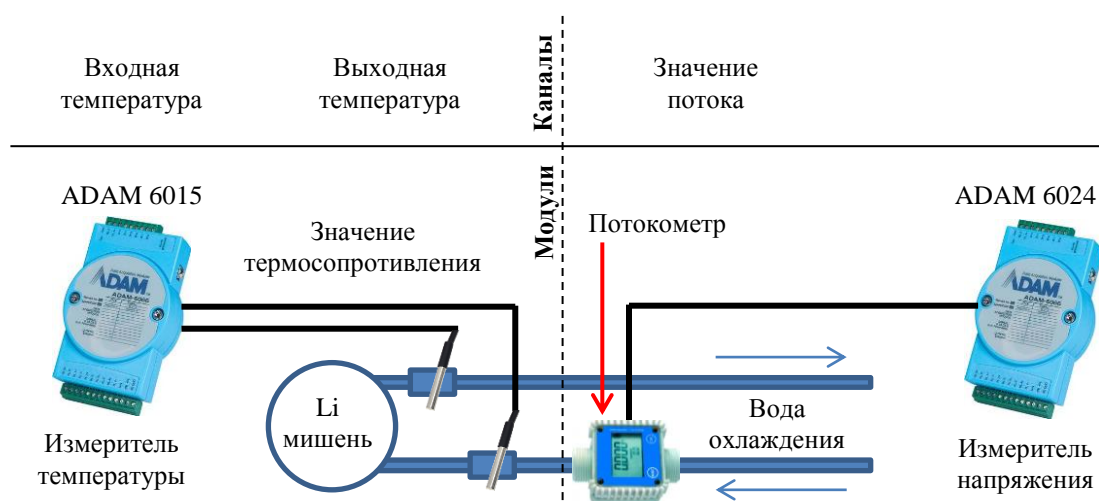


Рисунок 14 – Принципиальная схема измерителя мощности на Li мишени

Для измерения потока воды был использован потокомер Proteus 04004SN2 TPD с диапазоном измерения 0.8 – 9.5 литров / мин. Значение потока требуется снимать аналого-цифровым преобразователем (АЦП), где ноль вольт – минимальное значение потока, а десять вольт – максимальное значение потока. В качестве АЦП был выбран ADAM 6024.

Для диагностики напряжения были выбраны промышленные модули ADAM 6024 с интерфейсом связи Ethernet и протоколом Modbus. Они имеют шесть 16 битных АЦП, два 16 битных ЦАП, два ЦВх и два ЦВых.

Для измерения температур были выбраны промышленные модули ADAM 6015 с интерфейсом связи Ethernet и протоколом Modbus. Они имеют семь каналов измерения температуры RTD (resistance temperature detector). Также были использованы модули ICPCON Pet 7215 с точно такими же характеристиками для сравнения производителей по качеству. На ускорителе используются резисторы типа RT100 (ЧЭПТ-1) с трехпроводной схемой подключения. Схема подключения представлена на рисунке 15.

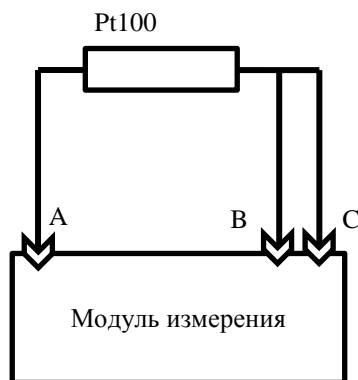


Рисунок 15 – Схема подключения RTD резисторов тремя проводами

Такая схема позволяет устанавливать измеритель и резистор на любое расстояние друг от друга. Устройство измерения производит замер сопротивления по формуле:

$$R = R_{AB} - R_{BC}.$$

Эта формула корректно работает только тогда, когда все три провода одинаковой марки и одинаковой длины. Такое включение используется для вычитания паразитного сопротивления проводов.

Формула расчета мощности приведена ниже:

$$P = \frac{cm\Delta T}{v} = IU,$$

$$P = \frac{C\rho lS(T_{\text{ВЫХ}} - T_{\text{ВХ}})}{v},$$

где

P – мощность,

C – удельная теплоемкость воды,

ρ – плотность воды,

l – длина трубы,

S – площадь сечения трубы,

$T_{\text{ВХ}}$ и $T_{\text{ВЫХ}}$ – значения температуры на входе и выходе,

v – скорость потока воды.

1.4.6 Ионный источник

В настоящее время проводятся эксперименты с ионным источником на специальном стенде. Для проведения эксперимента требуется запитать

поворотный магнит от промышленного источника Gwinstek PSU 6-200. Этот блок общается с консолью через Ethernet по протоколу Standard Commands for Programmable Instruments (SCPI). Требовалось управлять блоком дистанционно и считывать состояние устройства. Блок питания изображен на рисунке 16. Панель управления представлена на рисунке 17.



Рисунок 16 – Блок питания Gwinstek PSU 6-200

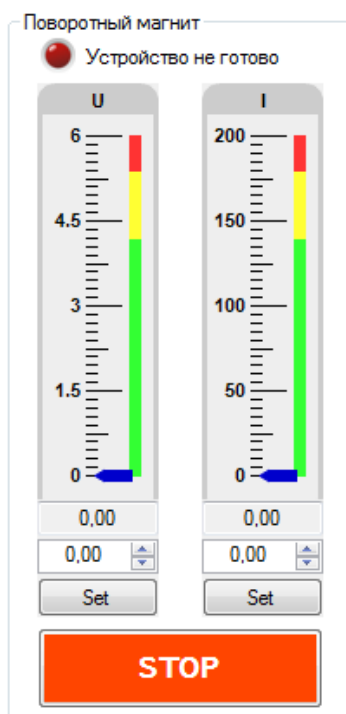


Рисунок 17 – Панель управления блоком питания

1.4.7 Ребутатор

Начальная конфигурация системы управления имела централизованную структуру. Это значит, что все сенсоры подключены к одной стойке измерения в центре зала. Из-за этого на длинных проводах образуются большие электромагнитные наводки, которые имеют импульсный характер, а они вызывают зависание оборудования. Ранее требовалось заходить в зал и перезагружать устройства вручную.

Но в процессе эксперимента радиационный фон может достигать единицы зиверт. Это означает, что за 10 минут человек получит примерно 200 годовых доз. Для примера на рисунке 18 изображена толщина двери входа в ускоритель.



Рисунок 18 – Толщина двери входа в ускоритель

Поэтому для возобновления работы системы было принято *временное* решение: подключить эти устройства к периферии с реле и дистанционно перезагружать их из пультовой. Проверка включения осуществляется пингованием устройства.

Данное решение очень некрасиво, но оно нужно для сохранения жизни персонала и обеспечения безопасности ускорителя в целом. В скором времени зависающее оборудование будет заменено на промышленное оборудование, и проблема будет устранена.

1.4.8 Тепловизор Flir T650SC

Для диагностики прохождения пучка через ускоритель используется тепловизор Flir T650SC. Фото тепловизора представлено на рисунке 19. Изображение с тепловизора представлено на рисунке 20.



Рисунок 19 – Тепловизор



Рисунок 20 – Фото из тепловизора

В настоящий момент интеграция изображения из тепловизора в программу управления обсуждается.

1.4.9 Пути развития системы

Дальнейшее развитие системы предусматривает интегрирование:

- Гамма и нейтронного детектора.
- Профилометра пучка по 9 точкам на вакуумном вводе движения.
- Бесконтактного измерителя тока.
- Вакуумной системы.
- Двух проволочного профилометра D-Race.
- Газового масс-спектрометра.
- Внедрение алгоритма сценариев из предыдущего диплома [5, 6].

1.5 Функциональные требования к системе управления

Контроль управляющих узлов должен осуществляться при помощи заложенных в них протоколов связи (например, Modbus). Диагностика состояния ускорителя осуществляется через измерение токов, напряжений и температур.

В качестве измерителей параметров были выбраны промышленные устройства ввода-вывода, управляемые программой оператора через Ethernet.

Система должна соответствовать следующим требованиям:

- Автоматически поддерживать обновления с сервера.
- Обладать длительной и стабильной работоспособностью.
- Работать в условиях постоянных пробоев и электромагнитных наводок.
- Система должна быть легко масштабируемой.
- Выводить данные на экран в удобной для оператора форме.
- Состоять из легко заменяемых модулей ввода / вывода.
- Записывать журнал всех измерений в базу данных.
- Обладать возможностью применения в других задачах.

2 АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧИ

2.1 Начальная конфигурация ускорителя

Изначально весь ускоритель был разработан из модулей, произведенных в ИЯФ СО РАН. Система управления проработала десять лет, что показывает ее надежность, но в настоящий момент она устарела. При попытке ее модернизировать или просто починить возникают проблемы и новые неполадки. В разработку этой системы было вовлечено три лаборатории и более пятнадцати человек. По истечении десяти лет некоторые сотрудники уже уволились, а другие – уже не помнят, как устроена система. Например, для измерения радиации используется старая система измерения, которая работает на операционной системе Windows 95, она изображена на рисунке 21.



Рисунок 21 – а) Компьютер с Windows 95; б) программа измерения радиации

Система разработана таким образом, что часть критичных узлов не была защищена от электромагнитных наводок, и из-за этого зависала.

Устаревшая система имеет централизованный характер. Это означает, что все измерения с тракта пучка приходят в стойку измерения в центре зала. При измерении напряжения на другом конце ускорителя при пробоях на длинном проводе образуются электромагнитные наводки, которые значительно уменьшают точность измерения, они могут также вывести АЦП из строя. Упрощенная схема расположения узлов показана на рисунке 22.

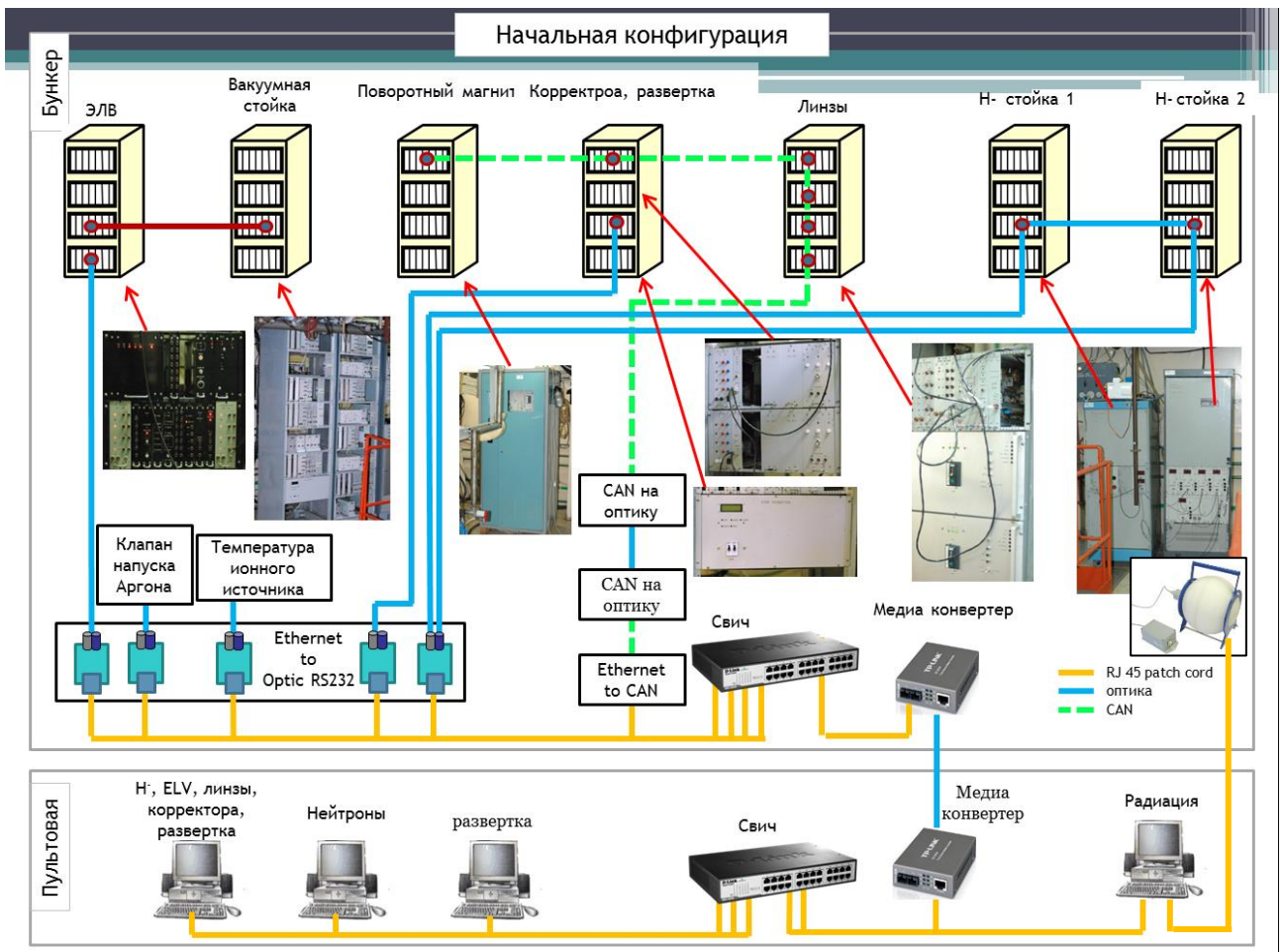


Рисунок 22 – Упрощенная схема расположения узлов устаревшей системы

В ходе развития установки приходилось многократно модифицировать интерфейс программы, что привело к визуальной перегрузке окна управления, что затруднило эксплуатацию установки. В качестве примера в устаревшей программе все еще указываются диагностики, которые были исключены пять лет назад. Было принято решение обновить архитектуру программы и системы управления на основании полученного опыта. Устаревшая программа управления приведена на рисунке 23.

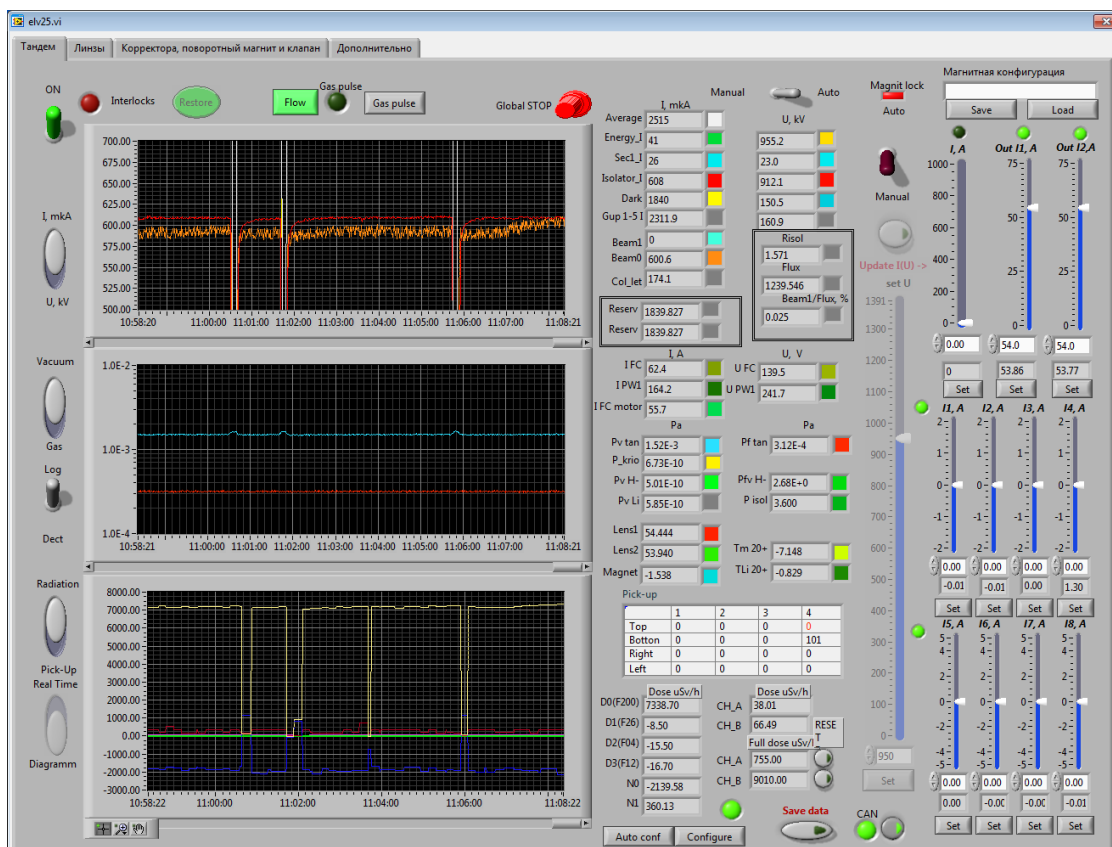


Рисунок 23 – Устаревшая программа управления

2.2 Предлагаемая конфигурация

Предлагаемая система управления имеет децентрализованный характер. Децентрализованная система управления – это система управления, у которой модули сбора данных расположены рядом с объектом измерения. При этом их много, и рядом с каждым объектом измерения стоит свой измеритель.

В процессе разработки системы было принято решение использовать промышленные устройства ввода/вывода и блоки питания, проверенные производителем. Общая структура измерения и контроля ускорителем на базе новой системы управления представлена на рисунке 24.

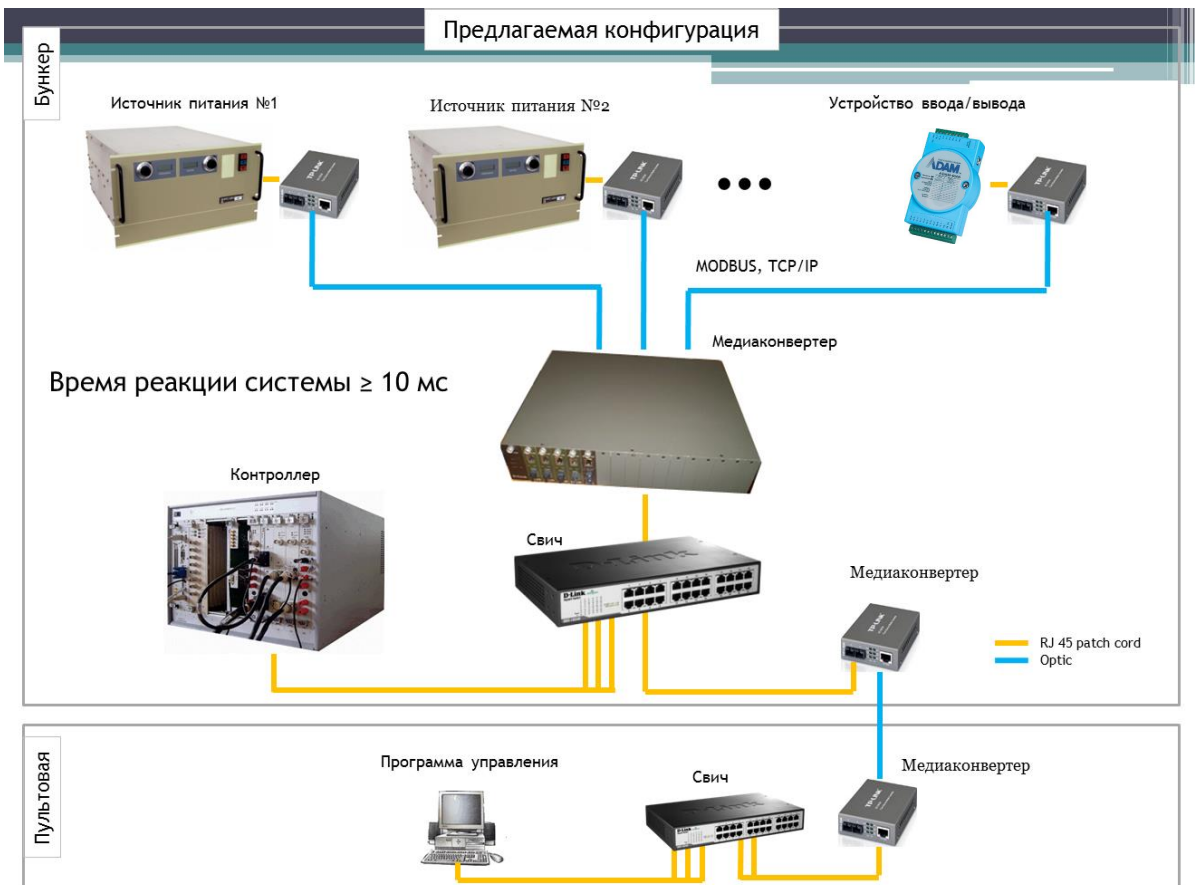


Рисунок 24 – Общая структура новой системы управления

В данной работе решается задача *создания и тестирования* новой системы управления. Внедрение пока что происходит только на части ускорителя, так как в настоящее время проводятся эксперименты, и времени на полную замену системы нет. Полная замена оборудования будет произведена только после серии тестов из-за соображений безопасности.

Программирование осуществлялось на языке «С#» с использованием следующих библиотек:

- графической библиотеки ChartView.
- Modbus.
- SCPI.
- SQL client.

В процессе создания программы управления была использована система контроля версий git, а также был создан личный репозиторий на RaspberryPi.

2.3 Конфигурация программы

Созданная программа поддерживает возможность считывать конфигурацию из файла. В качестве формата конфигурационного файла был выбран XML. Это позволило хранить конфигурацию в иерархической структуре и предоставить файл в удобном для оператора виде. Пример конфигурационного файла приведен в приложении А.

В случае выхода какого-нибудь устройства из строя оператору достаточно переподключить измерение на другой модуль в свободный канал, подправить конфигурацию и продолжить работу. Обученному оператору потребуется *не более 10 минут* на переконфигурирование системы без перекompиляции программы!

2.4 Ведение журнала

2.4.1 Общий принцип

Запись журнала ведется локально на компьютер в Excel-файл и в SQL базу данных, настроенной на мини-компьютере RaspberryPi. Запись файлов в Excel-файл ведется из-за исторических соображений, так как иногда физикам проще взять Excel-файл и работать с ним, чем делать запрос в SQL. Поэтому в настоящий момент ведется разработка утилиты выборки данных из базы данных и конвертация их в файл Excel. Для этого была создана библиотека LogWizard, которая описана в п. 4.2

Кроме измерений, в журнал регистрируются сообщения о текущем состоянии системы, которое отображается внизу экрана на панели оператора. Пример сообщений представлен на рисунке 25.

Time	Text
12:52:25	Устройство 'Ребуталка' подключено
12:52:25	Устройство 'Пищалка' подключено
12:52:25	Устройство 'Шиберы' подключено
12:52:26	Устройство 'Пирометр' подключено
12:57:38	Связь с модулем 'Охлаждаемая диафрагма и после магнита' потеряна из-за ошибки 'Не удастся прочитать данные из транс
12:57:40	Связь с модулем 'Охлаждаемая диафрагма и после магнита' восстановлена'

Рисунок 25 – Окно состояния системы управления

2.4.2 Ведение журнала в устаревшей программе

Так как на ускорителе все-еще используется устаревшая программа, требуется записывать ее данные в ту же базу данных, в которую пишет новая программа управления.

Для решения этой задачи первым делом открывался файл с журналом устаревшей программы. Потом этот файл анализируется и отправлялся в базу данных. Однако когда старая программа записывала данные, а программа-анализатор пытается прочитать – возникает конфликт ресурсов. В этот момент старая программа выводила сообщение об ошибке и зависала, а из-за этого ускоритель оставался без контроля на неопределенный срок.

Однако появилось понимание, как сделать чтение данных из устаревшей программы без конфликта записи в этот же файл. Это можно достичь путем открытия файла с параметром `fmShareDenyNone`.

2.4.3 Перехват ошибок в программе

Для удобства отладки программы был разработан механизм сохранения лога о возникшей ошибке на компьютер. После сохранения лога происходит отправка письма на электронный адрес разработчика с описанием проблемы. Это позволяет разработчику всегда быть в курсе текущего состояния программы и с ускорителя.

3 АРХИТЕКТУРА ПРОГРАММЫ УПРАВЛЕНИЯ

Для масштабируемости системы была разработана архитектура на парадигме объектно-ориентированного программирования (ООП). Основная идея заключается в том, что разделить работу на три уровня:

1. Устройства.
2. Каналы.
3. Модули.

Модули – это устройства ввода / вывода.

Устройства – совокупность модулей, объединенных для диагностики и управления определенного узла ускорителя.

Канал – связующая часть между модулем и устройством.

В качестве примера рассмотрим измеритель мощности (подробнее в п. 1.4.5). Это устройство, которое измеряет мощность P_i мишени по входной температуре, выходной температуре и потоку воды. После измерения параметры рассчитываются по формуле, и оператору выводится только рассчитанная мощность. Однако при желании всегда можно посмотреть «сырые данные». Логическая схема измерителя мощности представлена на рисунке 26.

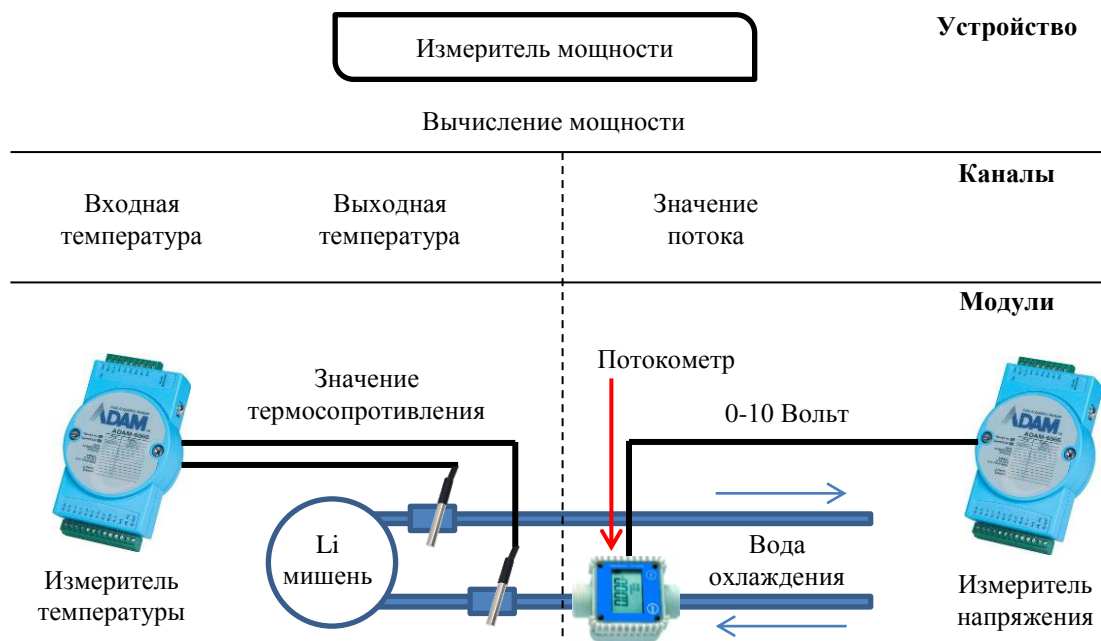


Рисунок 26 – Логическая схема измерителя мощности

Идея в том, что устройству не важно, откуда и каким образом считываются данные. Можно в любой момент начать считывать значения

температур с двух разных модулей, но с точки зрения устройства ничего не поменяется. Пример эквивалентной системы представлен на рисунке 27.



Рисунок 27 – Эквивалентная схема измерителя мощности

На рисунке 26 и рисунке 27 изображены эквивалентные схемы. Для уровня устройства ничего не изменилось: как были каналы температуры и напряжения, так и остались. За сбор данных отвечает каналный уровень. Он опрашивает все модули и передает информацию на уровень устройства. Каждый уровень детальнее описан в следующих пунктах.

3.1 Устройства

Устройства – совокупность модулей, объединенных для диагностики и управления определенного узла ускорителя. Этот узел может состоять из множества разных модулей, но предназначение у него одно.

Примерами могут быть: пирометр, шиберы, охлаждаемые диафрагмы и прочее (подробнее в п 1.4). Иногда устройство может состоять из вложенных устройств. Такая возможность была заложена для объединения схожих узлов в одну категорию.

3.2 Каналы

Каналы – связующее звено между устройством и модулем. В канале хранится информация об отображаемом имени, имени для колонки логов, а

также цвет на графике. Эти данные считываются из конфигурационного файла, пример которого описан в приложении А.

При описании модуля в библиотеке модулей указывается его функционал. Это делается по средствам реализации интерфейсов. В процессе описания модуля указывается лишь требуемый интерфейс.

Интерфейс – это регламент взаимодействия. Класс, который реализует интерфейс, обязан реализовывать все его методы. Например, у интерфейса реле будут объявлены функции чтения, функции записи и массив, в котором хранятся значения реле.

Таким образом, было создано шесть интерфейсов:

- АЦП.
- ЦАП.
- ЦВх.
- ЦВых.
- Реле.
- Температура.

При этом возможность записи есть только у ЦАП, ЦВых и реле. Можно было бы еще добавить температуру, но на практике установка температуры еще ни разу не требовалась. Возможность считывания данных есть у всех каналов по умолчанию.

Удобство каналов заключается в том, что при описании канала указывается требуемый интерфейс, а при чтении конфигурации происходит проверка модуля на наличие требуемого интерфейса. Это означает, что можно выбирать любой модуль, у которого реализован требуемый функционал.

Таким образом, в случае выхода какого-нибудь устройства из строя оператору достаточно переподключить измерение на другой модуль, реализующий нужный интерфейс, подправить конфигурацию и продолжить работу. Обученному оператору потребуется *не более 10 минут* на переконфигурирование всей системы без перекомпиляции программы!

Каналы могут быть двух типов: бинарные и вещественные. Они логически разделены на виртуальные и реальные. Их различия будут описаны в следующих пунктах. Диаграмма классов представлена на рисунке 28.

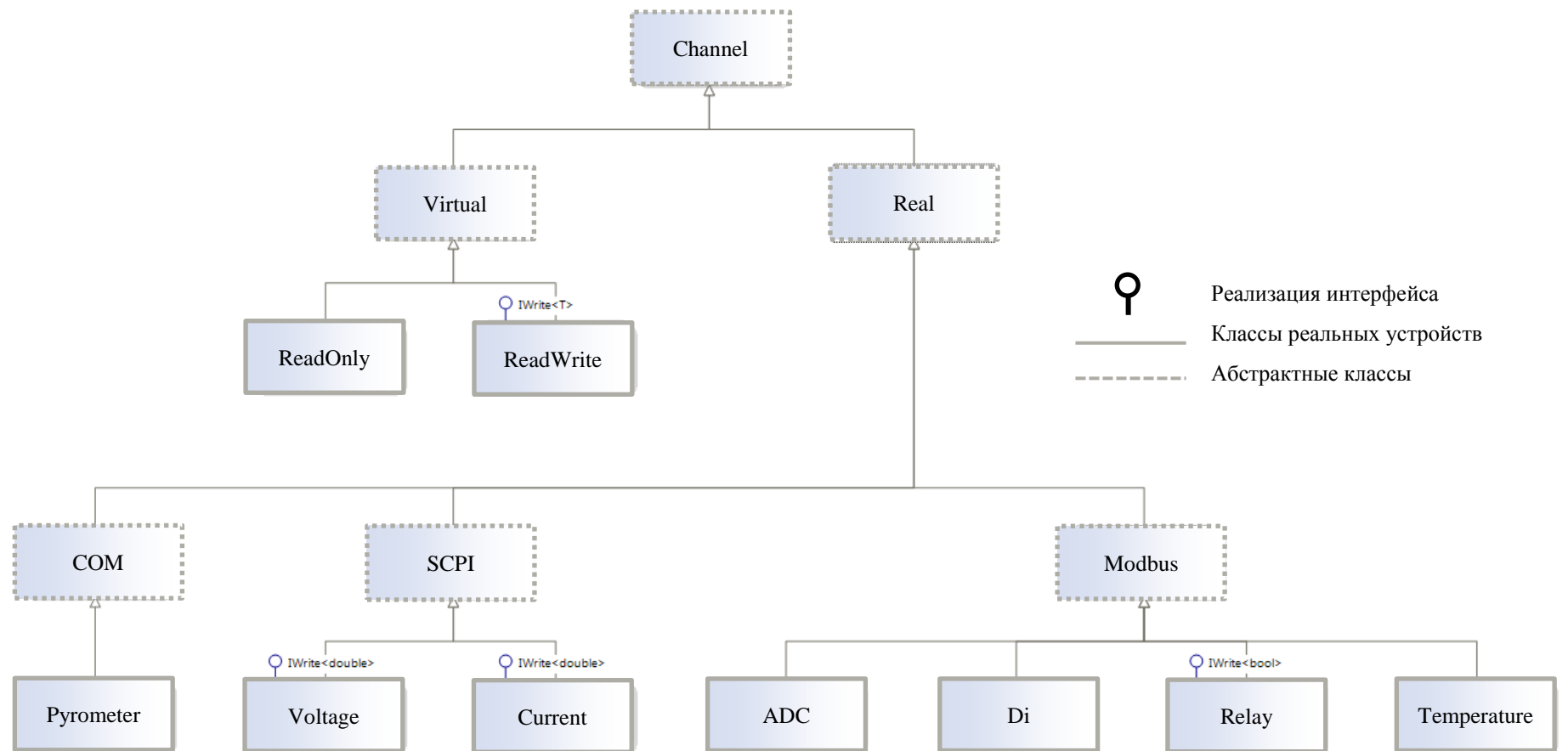


Рисунок 28 – Диаграмма классов

3.2.1 Реальные каналы

Реальные каналы – это каналы, которые связаны с реальным устройством, таким как устройство ввода / вывода или блок питания.

3.2.2 Виртуальные каналы

Виртуальные каналы – это каналы, которые не связаны с конкретным модулем. Это канал, значение которого рассчитывается по заданной формуле.

В качестве примера рассмотрим потокометр Proteus 04004SN2 TPD. Требуется снимать аналого-цифровым преобразователем (АЦП), где ноль вольт – минимальное значение потока, а десять вольт – максимальное значение потока. Соответственно, значение на канале АЦП будет от 0-10 В, что для оператора может быть не понятно и не нужно, так как ему интересен поток.

Для решения этой задачи можно создать виртуальный канал и указать у него функцию перерасчета из напряжения в поток, и именно ее выводить на экран, при этом все значения (сырые и рассчитанные) пишутся в журнал.

Также виртуальный канал используется в измерителе мощности (подробнее в п. 1.4.5). В нем мощность рассчитывается на основании двух температур и потока воды в системе охлаждения. Сложность функции расчета не ограничена.

3.3 Модули

Модули – устройства ввода / вывода, которые выполняют установку / измерение таких величин, как температура, напряжение, ток. Основная идея заключается в том, что на каждое устройство создается свой поток, в котором происходит опрос всей периферии и сохранение полученных данных в буфер. Далее каналный уровень передает информацию в устройство.

Для модулей была разработана библиотека ModuleWizard (подробнее в п. 4.1). В разработанной библиотеке были созданы устройства с заданным в них функционалом, посредством интерфейсов (подробнее в п. 3.2).

В общем случае примером модулей могут быть блоки питания или устройства удаленного ввода / вывода.

3.4 Утилиты

Для решения некоторых задач потребовалось создания еще одного уровня. Он не связан с предыдущими, так как не использует модули.

До сих пор единственным применением модулей была задача расчета флюенса. Флюенс – интеграл по времени от плотности потока частиц. Этот параметр очень важен, когда мы проводим совместный эксперимент на тему блистеринга с коллегами из Японии.

Реализация была построена на анализе логов устаревшей программы, которая раз в секунду добавляла значение всех каналов в Excel-файл. Для решения этой задачи был написан алгоритм, который по заданной дате искал файлы, кешировал их для быстроты дальнейшей работы, а после считал интеграл по току на мишени.

В первых версиях программы флюенс считался автоматически каждую минуту, но возникла проблема считывания логов с устаревшей программы. Перед изучением файла они сначала копируются во временную папку и только потом кэшируются и анализируются. Во время копирования файла он блокируется операционной системой, и программа оператора вылетает с ошибкой, что недопустимо. Эта ошибка была решена путем открытия файла с параметром `fmShareDenyNone`.

3.5 Разработанная программа управления

Программа управления выполняет функцию сбора данных с модулей ввода / вывода, управления ими по запросу оператора и сохранение измерений в базу данных. Из-за того, что новая система пока что заменила только часть устаревшей системы, программа оператора имеет ограничения по размерам окна. Поэтому программа была разбита на следующие логические формы:

- Основное.
- Управление.
- Флюенс.
- Ребутатор.
- Источник Н.
- Все каналы.

Подробнее о каждом разделе будет рассказано в следующих разделах. Листинг окна программы оператора приведен в приложении Б.

3.5.1 Форма «Основное»

На данной форме показывается общее состояние ускорителя, а именно: состояние подключения всех устройств, температуры на охлаждаемых диафрагмах и пирометра. Эта форма приведена на рисунке 29.

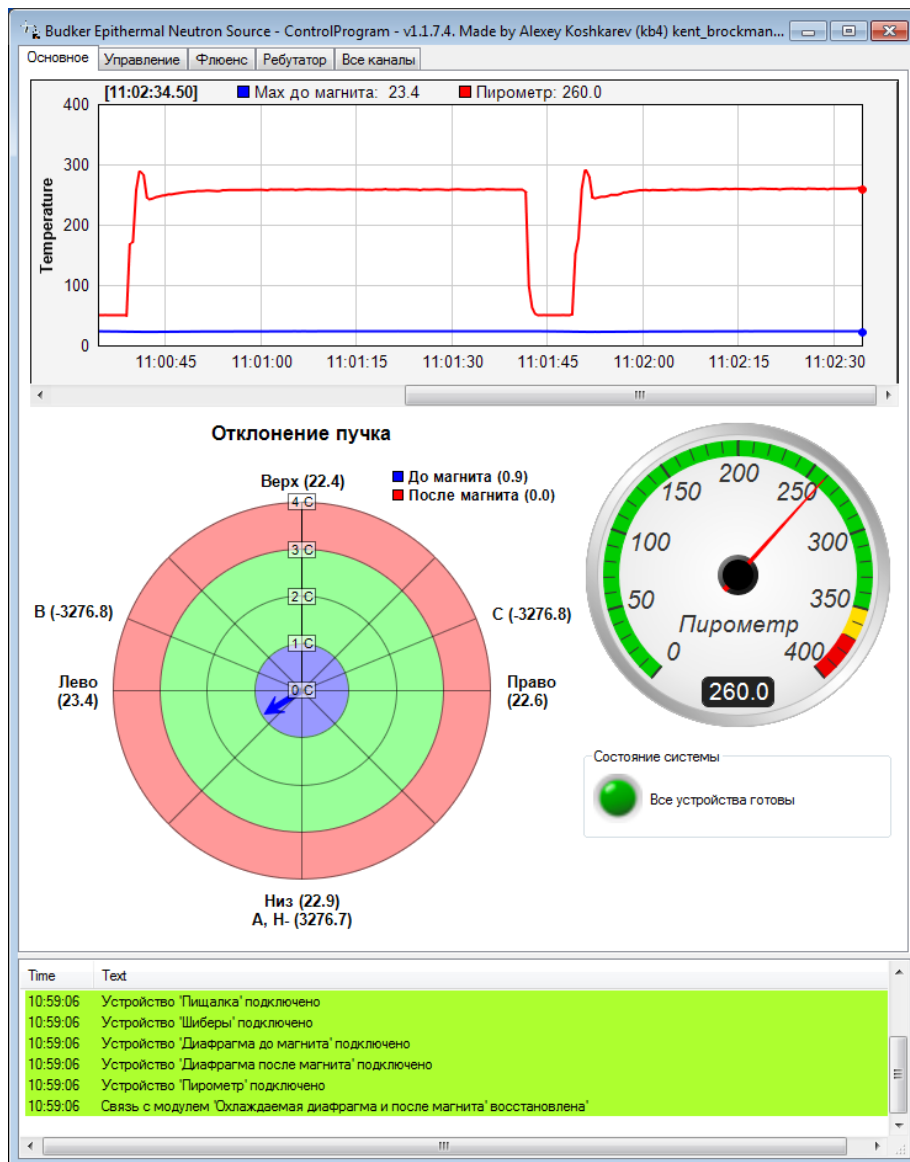


Рисунок 29 – Форма «Основное»

3.5.2 Форма «Управление»

На данной форме предоставлен интерфейс управления узлами, такими как: вакуумные шиберы и блок питания супрессора вторичной эмиссии. Подробнее об этих устройствах рассказано в п. 1.4.3 и п. 1.4.4 соответственно. Эта форма приведена на рисунке 30.

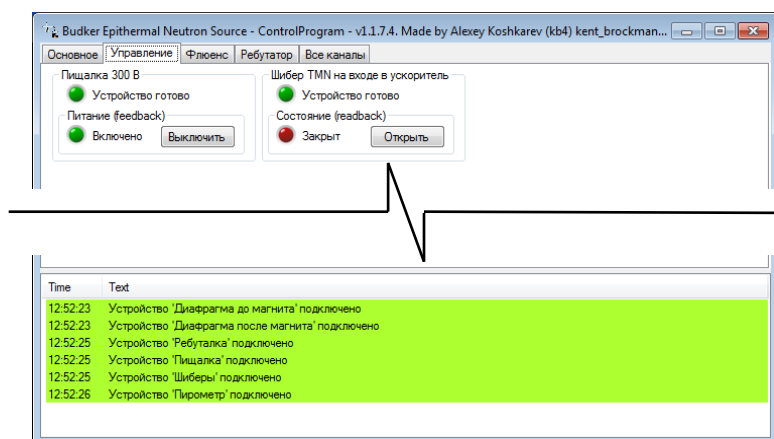


Рисунок 30 – Форма «Управление»

3.5.3 Форма «Флюенс»

На данной форме предоставлен интерфейс расчета флюенса за произвольный период. Флюенс – интеграл по времени от плотности потока частиц. Этот параметр очень важен в процессе проведения экспериментов над блистерингом. Подробнее данный функционал описан в п. 3.4. Эта форма приведена на рисунке 31.

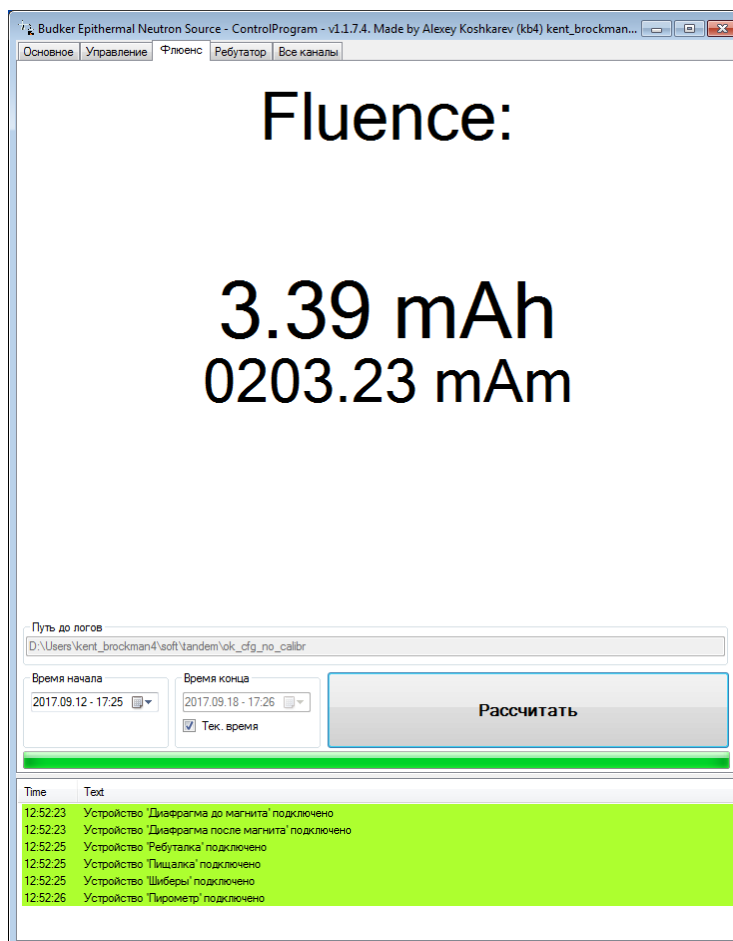


Рисунок 31 – Форма «Флюенс»

3.5.4 Форма «Источник Н»

На данной форме предоставлен интерфейс работы с блоком питания поворотного магнита ионного источника (Н). Изначально планировалось разместить данный функционал на форме управления, но этот источник временно будет стоять на специальном стенде, на котором весь функционал ускорителя не нужен. Оператор будет управлять только источником питания. Чтобы не загромождать интерфейс не нужным в данный момент функционалом, было принято решение разместить интерфейс работы с блоком

питания на отдельную вкладку. Подробнее данный функционал описан в п. 1.4.6. Эта форма приведена на рисунке 32.

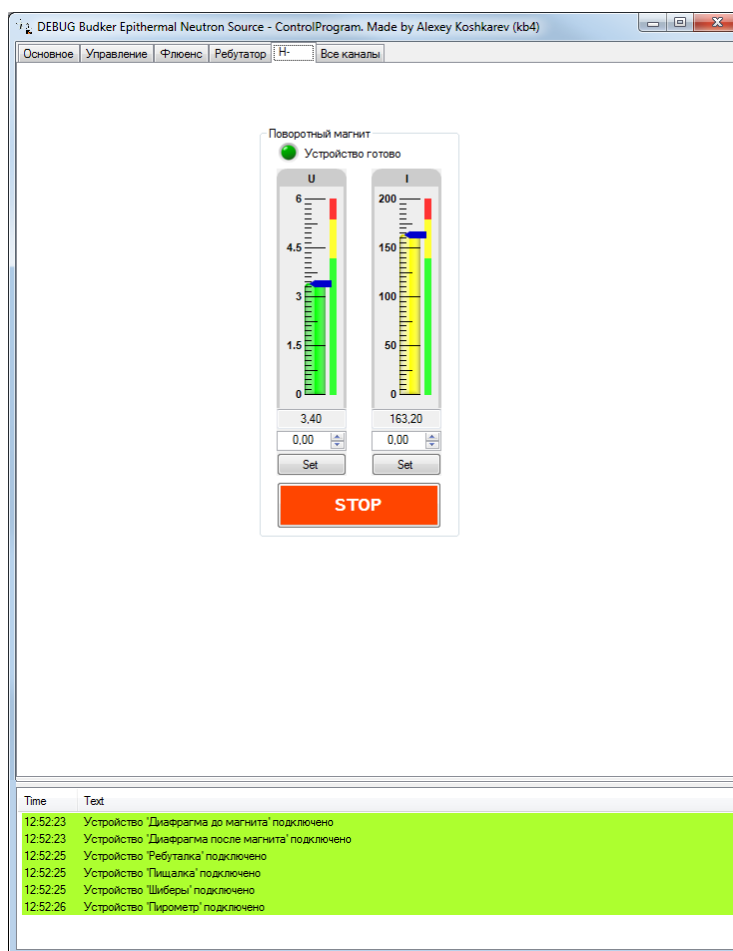


Рисунок 32 – Форма источник «H»

3.5.5 Форма «Ребутатор»

Данная форма является временной мерой для восстановления работы ускорителя посредством перезагрузки нескольких устройств. Подробнее данный функционал описан в п. 1.4.7. Эта форма приведена на рисунке 33.

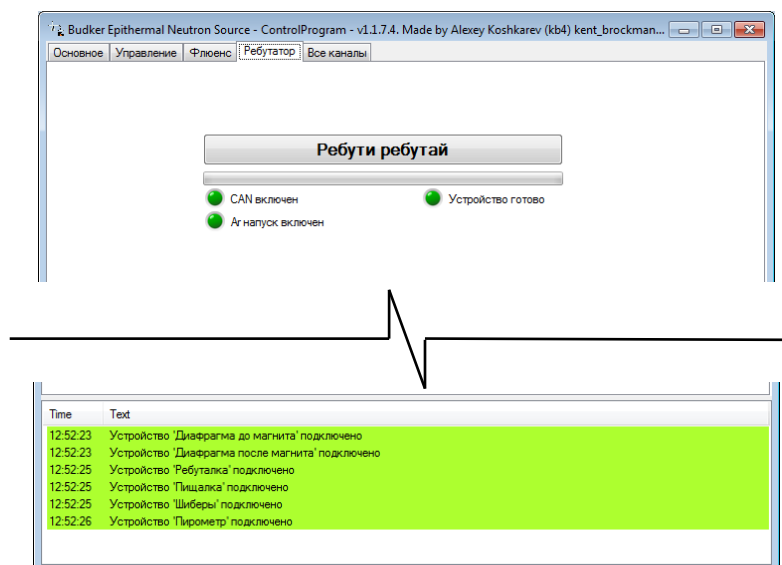


Рисунок 33 – Форма «Ребутатор»

3.5.6 Форма «Все каналы»

На данной форме оператор может изучить «сырые данные» со всех устройств и каналов. Данный функционал очень полезен ученым для самостоятельного оценивания показаний измерителей. Эта форма приведена на рисунке 34. Этот интерфейс разработан, в том числе, для отладки. По двойному клику можно вручную изменить значение канала на устройстве. Пример изменения значения указан на рисунке 35. При попытке изменения значения на канале, у которого нет возможности записи, система сообщит об ошибке с описанием проблемы.

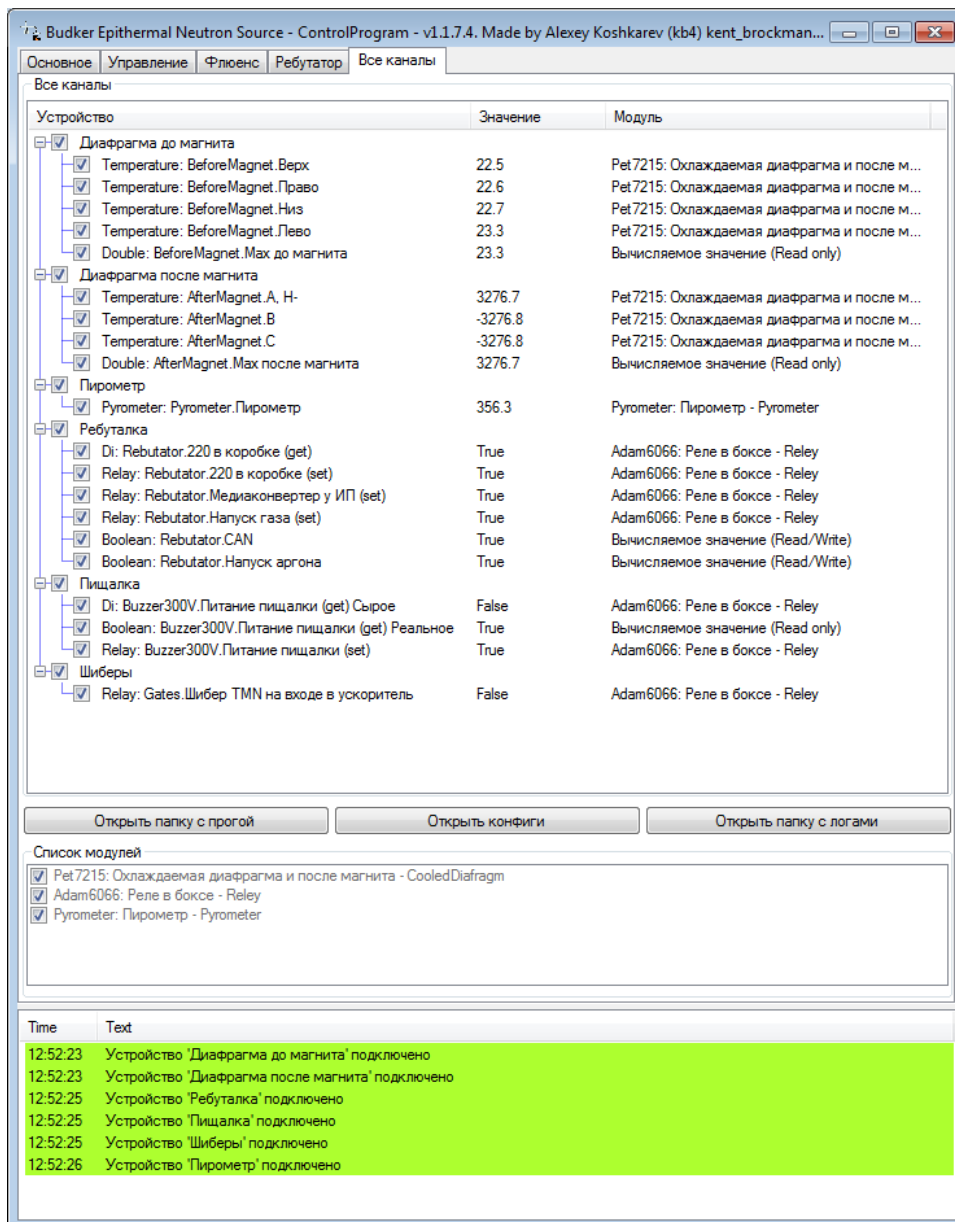


Рисунок 34 – Форма «Все каналы»

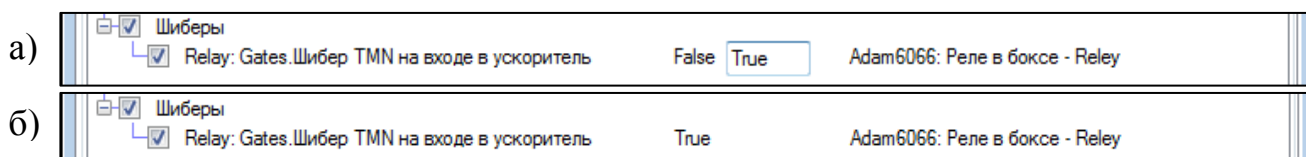


Рисунок 35 – Ручное изменение значение канала на устройстве: а) перед изменением; б) после изменения

4 РАЗРАБОТКА БИБЛИОТЕК

4.1 ModuleWizard

4.1.1 Общий принцип

Библиотека ModuleWizard была разработана в процессе создания программы. Задача этой библиотеки – описать все модули управления и предоставить функционал, позволяющий в удобном для программиста виде использовать модули внешней периферии.

В первую очередь были описаны модули внешней периферии ADAM и ICP CON. После был реализован пирометр, а далее источник питания PSU 6 200. На рисунке 36 изображена схема классов и реализации интерфейсов.

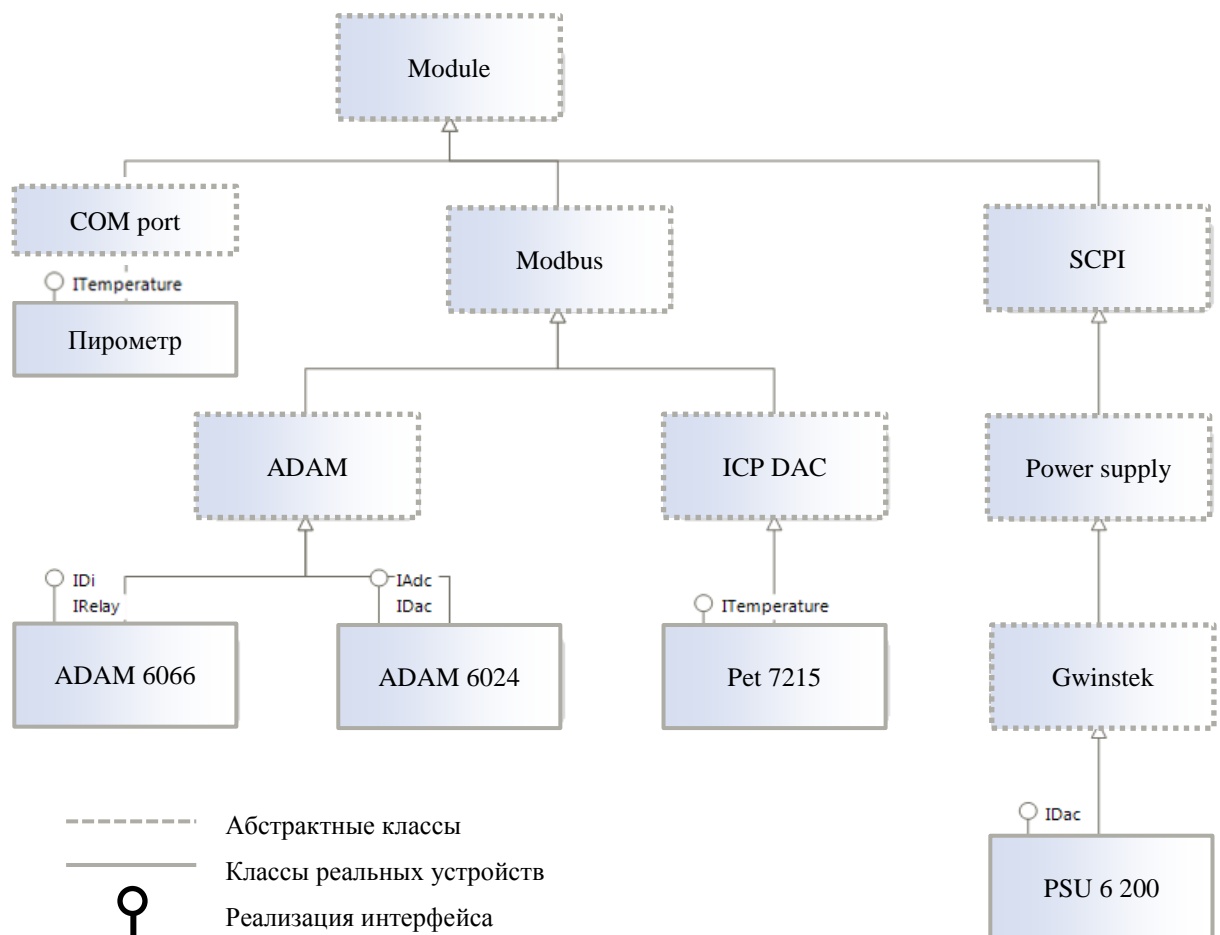


Рисунок 36 – Схема классов

В классе *Module* описаны основные принципы работы с любыми устройствами.

Сначала происходит подготовка потока: присваивание имени для отладки, создание переменных в потоке устройства. Далее совершаться

попытка подключения к модулю. При успешном подключении первым делом производится считывание выходных регистров устройства, а потом в бесконечном цикле происходит сбор данных и, при необходимости, установка пользовательских значений. Схема работы процесса приведена на рисунке 37.

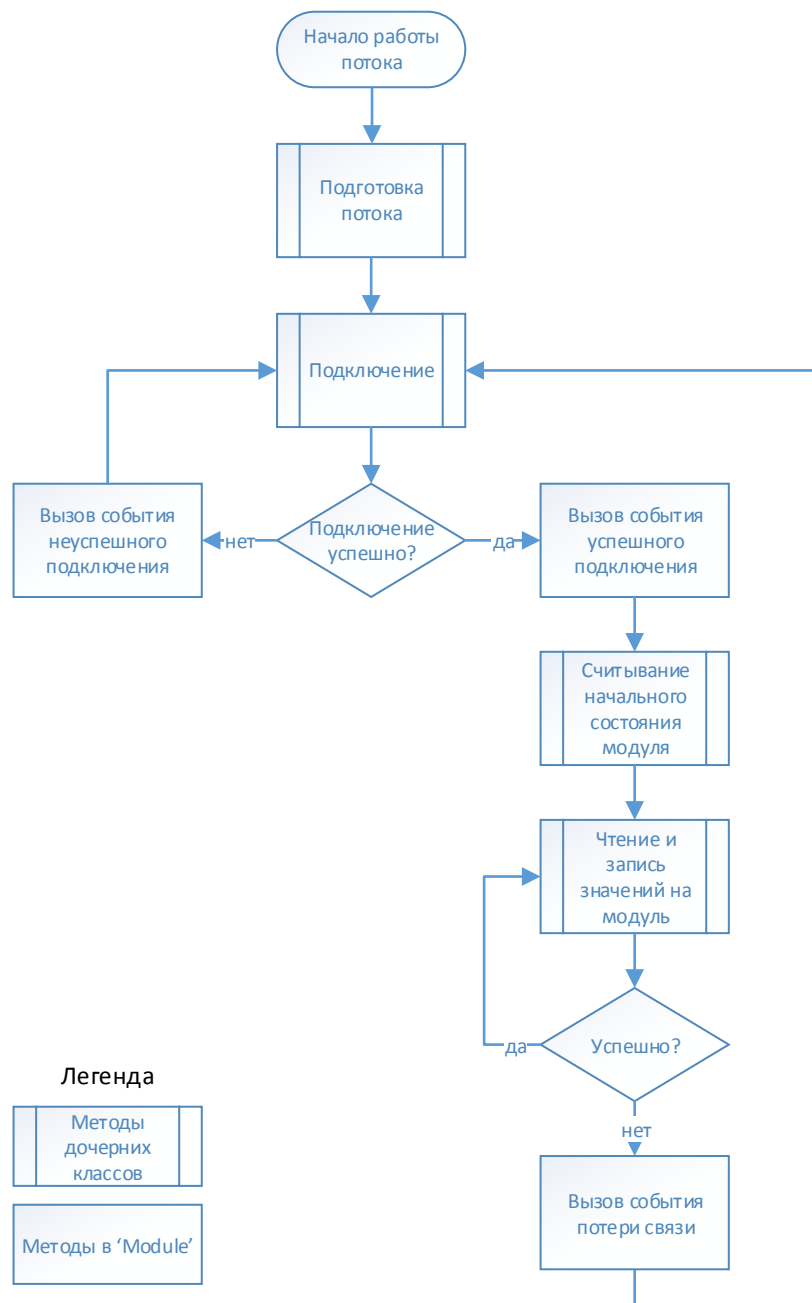


Рисунок 37 – Схема работы процесса модулей

Удобство библиотеки заключается в том, что методы «Подготовка потока», «Подключение», «Считывание начального состояния модуля» и «Чтение и запись значений на модуль» определяются в дочерних классах, которые могут иметь многоуровневую структуру. На каждом уровне

реализуются определенные методы этого уровня. Это сделано для обобщения логики работы с устройствами.

Листинг класса *Module* приведен в приложении В.

4.1.2 Пример

В качестве примера отдельно рассмотрим ветку Modbus, которая приведена на рисунке 36.

На уровне Module описаны общие принципы работы модулей.

На уровне Modbus описано подключение к устройству по протоколу Modbus и функции проверки статуса подключения (подключен / не подключен). Также на этом уровне описаны функции чтения / записи в устройство по протоколу Modbus.

На уровне Adam и IscDac описаны функции перерасчета «сырых» значений АЦП / ЦАП в реальное напряжение и температуру. У каждого производителя функции пересчета разные.

На уровне Adam6066, Adam6024 и Pet7215 описан функционал устройства, то есть реализация интерфейсов. Также в каждом устройстве в каждом типе периферии (АЦП, ЦАП, Температура) описываются адреса, по которым хранятся данные о модуле. Например, у Adam6066 данные о реле начинаются с 0x10, а ЦВх с 0x00. Такая же структура у остальных ветвей. Класс Adam6066 приведен в приложении Г.

4.2 LogWizard

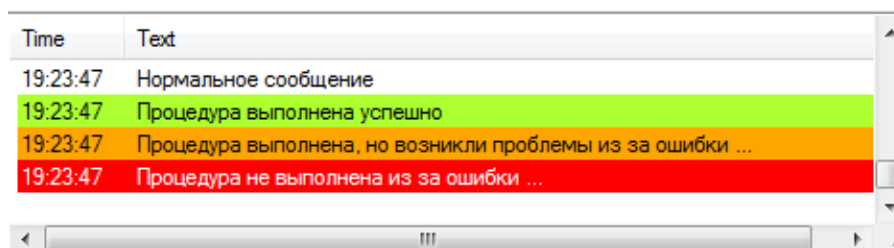
Библиотека LogWizard была разработана в процессе создания программы управления. В этой библиотеке были собраны воедино механизмы отображения и записи логов в файл и базу данных. Разработанные компоненты описаны ниже.

4.2.1 FormLog

Компонент FormLog был разработан для отображения оператору общего состояния системы. Иногда недостаточно обычного индикатора, который может отобразить два или три состояния, а требуется вывести сообщение об ошибке с ее описанием.

Удобство заключается в том, что при выводе сообщения указывается тип сообщения (хорошо, предупреждение, плохо) с соответствующим цветовым

оформлением и временем вывода сообщения. Пример использования представлен на рисунке 38.



Time	Text
19:23:47	Нормальное сообщение
19:23:47	Процедура выполнена успешно
19:23:47	Процедура выполнена, но возникли проблемы из за ошибки ...
19:23:47	Процедура не выполнена из за ошибки ...

Рисунок 38 – Пример использования компонента FormLog

4.2.2 TimeLog

Компонент TimeLog был разработан для сохранения журнала измерений в Excel-файл. В первых версиях была использована библиотека для работы с Excel файлами от Microsoft. Но из-за ограничения по количеству строк в таблице была разработана собственная библиотека которая решала эту проблему. Разработанная библиотека не требует частого обращения к жесткому диску, а пишет данные блоками через заданный интервал времени.

В программе пользователь (программист) лишь вызывает функцию записи в буфер данных. После заполнения буфера пользователь отправляет команду на сохранение строки данных. После многократного повторения этой операции образуется массив строк. Библиотека настроена таким образом, что запись массива строк на диск происходит автоматически раз в минуту (параметр может быть изменен пользователем). Это позволило значительно разгрузить жесткий диск, и в случае вылета программы сохранятся все данные, кроме последних 60 секунд, что является допустимым для эксперимента.

Массив строк пишется в текстовый файл с расширением xlsx. Формат строки такой, что данные пишутся парами: строковое представление данных, а потом знак табуляции, и так со всеми параметрами. При открытии данных журнала Excel'ем, они успешно открываются и распределяются по колонкам автоматически.

4.3 GraphWizard

Библиотека GraphWizard была разработана в процессе создания программы управления. В этой библиотеке реализовано отображение графической части интерфейса. В данной библиотеке за основу берутся компоненты из графической библиотеки ChartDir.

Проблема использования ChartDir состоит в том, что для реализации компонента требуется написание большого количества кода, а если компонентов несколько, то читаемость программы резко падает, что снижает качество поддержки программного продукта.

В первых версиях библиотеки был описан класс, в конструкторе которого указывалось полотно ChartDir, которое требовалось разместить на форме заранее, а также параметры отображения. Например, у Led индикатора эти параметры такие: текст при успехе, предупреждении и ошибки. Это неудобно, так как полотно не очевидно связан с классом Led.

Для решения этой задачи был разработан компонент WinForms, внутри которого было определено полотно, методы отображения и стандартные для компонента поля позиции, габаритов и прочее. Например, у компонента BarReadWrite (подробнее в п. 4.3.2) внешними полями были Min, Max, Avarage1, Avarage2 и Value. При изменении Value происходила автоматическая обновление компонента. Также в этом компоненте было реализовано событие при нажатии на клавишу Enter. Все компоненты наследуются от Control'a, так что поля x, y, width, height и пр. реализованы автоматически.

В данной библиотеке были разработаны компоненты, которые описаны ниже.

4.3.1 Компонент «Векторный радар»

Этот компонент был создан для отображения отклонения пучка. У этого компонента задача отобразить температуру на каждом температурном канале в градусах Цельсия и вектор отклонения, который задается в полярной системе координат, то есть углом и длиной. Компонент изображен на рисунке 39. Также на графике изображена легенда и длина вектора.

В настоящее время измеряется отклонение в двух местах, так что рисуются два вектора.



Рисунок 39 – Панель отклонения пучка: а) значение с охлаждаемой диафрагмы; б) значения после поворотного магнита

4.3.2 Компонент «Шкала значения»

Этот компонент был создан для ввода-вывода числовых данных и наглядного отображения значений в пользовательском интерфейсе. Этот компонент изображен на рисунке 40.

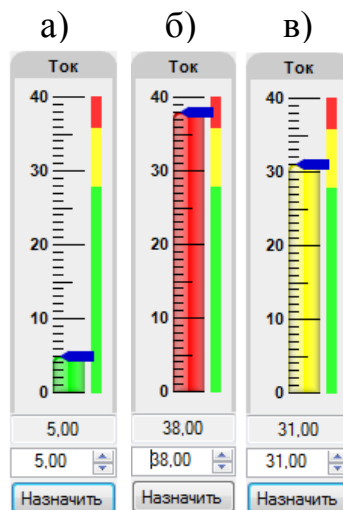


Рисунок 40 – Компонент шкала значения в трех состояниях: а) нормальное; б) предупреждение; в) опасно

Компонент имеет вид колонки, которая заполняется от минимума до максимума. Он может быть в трех состояниях:

- Зеленый – все в норме.
- Желтый – предупреждение.
- Красный – опасно.

Управление и диагностика ионного источника производится описанным компонентом.

4.3.3 Компонент «Лампочка»

Компонент имеет вид лампочки, что должно быть интуитивно понятно пользователю. Внешний вид индикатора представлен на рисунке 41.

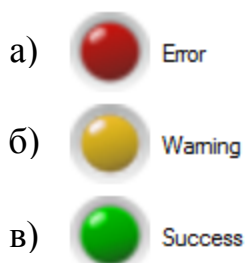


Рисунок 41 – Компонент лампочка в трех состояниях: а) нормальное; б) предупреждение; в) опасно

Компонент может быть в трех состояниях:

- Зеленый – все в норме.
- Желтый – предупреждение.
- Красная – ошибка.

Для каждого состояния можно настроить отдельный текст. Также доступно два размера индикатора.

4.3.4 Компонент «Стрелочный индикатор»

Компонент имеет вид стрелочного индикатора (спидометра). Данный вид отображения позволяет пользователю на интуитивно понятном уровне понимать, является ли значение в пределах нормы или нет. Этот компонент изображен на рисунке 42.

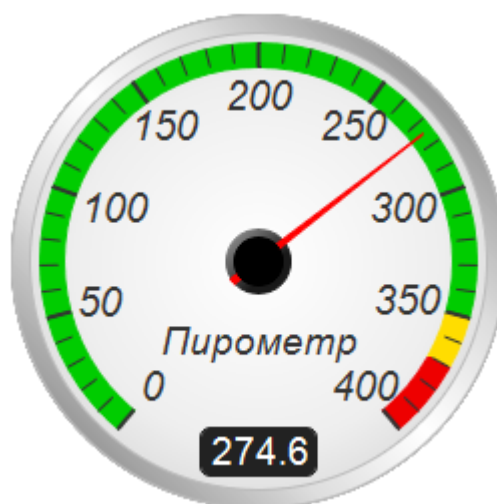


Рисунок 42 – Компонент спидометр

Индикатор имеет три зоны значения:

- Зеленая – все в норме.
- Желтая – предупреждение.
- Красная – опасно.

4.3.5 Компонент «График реального времени»

Данный компонент предназначен для отображения данных в реальном времени. Компонент позволяет выводить несколько серий данных. Разработанный компонент позволяет масштабировать график и менять время начала и конца. Компонент позволяет показать данные на определённый момент времени. При наведении курсора сверху указываются значения. Этот компонент изображен на рисунке 43.

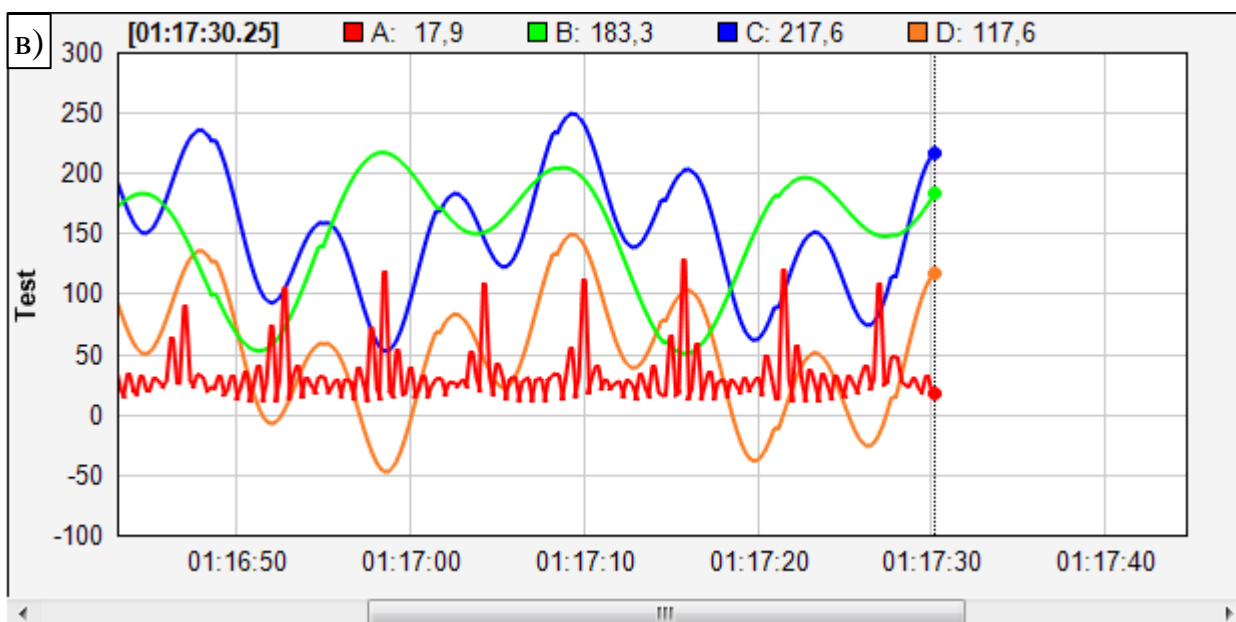


Рисунок 43 – График реального времени: а) период две минуты; б) период час; в) несколько серий данных

5 ОТЛАДКА И ТЕСТИРОВАНИЕ

Программа была проверена на отказоустойчивость. В течение трех дней подряд происходил сбор данных с трех узлов измерения. По результатам проверки сбоев не обнаружено.

Для проверки библиотек на предмет ошибок входных данных были разработаны модульные тесты.

В последних версиях библиотеки ModuleWizard обнаружилось, что одно устройство некорректно считывает данные при переподключении. В настоящее время ведется отладка модуля.

В старых версиях программы журнал велся в файл Excel в формате xls. Как оказалось, в этом формате было ограничение по количеству строк. Их количество не должно превышать 65536, а при превышении этого числа программа останавливалась с критической ошибкой из-за поставляемой библиотеки. Эта проблема была решена созданием собственной библиотеки ведения журнала в Excel-файл.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы была разработана адаптивная система автоматизации ускорительного источника эпитепловых нейтронов для БНЗТ на базе промышленного оборудования.

Система автоматизации была внедрена в ускоритель (имеется акт о внедрении) и начался этап отладки. Успешно проведено тестирование системы в условиях электромагнитных помех на протяжении десятков экспериментов.

Были поставлены и решены следующие задачи:

- Получен опыт работы по изучению сложных комплексов и систем.
- Изучены принципы работы с периферийными устройствами.
- Разработана программа управления.
- Налажена связь программы управления с устройствами ввода/вывода.
- Интегрировано автоматическое обновление программы с сервера.
- Получен опыт отладки многопоточных приложений.
- Разработана база данных для хранения данных об эксперименте.
- Разработанная система автоматизации может быть использована для решения любых задач, связанных с автоматизацией устройств.

В дальнейшем планируется расширять список устройств, управляемых новой системой управления и разработать язык автоматического управления ускорителем, как это было сделано в предыдущей дипломной работе [5, 6].

Таким образом, удалось создать систему управления для ускорителя, которая в дальнейшем позволит перейти от физических и биологических исследований в области БНЗТ к терапии больных злокачественной опухолью.

Выражаю искреннюю благодарность моему научному руководителю Таскаеву Сергею Юрьевичу за помощь в подготовке данной работы, Зубареву Петру Васильевичу за помощь в изучении ускорителя и всем сотрудникам лаборатории БНЗТ за поддержку.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Taskaev S.Yu. Accelerator based neutron source for the neutron-capture and fast neutron therapy at hospital. / S.Yu. Taskaev [at al] // Nuclear Instr. and Methods in Physics Research. – 1998. – No. A 413/2-3. – P. 397-426.
- 2 Taskaev S.Yu. Accelerative source of epithermal neutrons. / S.Yu. Taskaev // Phys. Element. Particles and Atomic Nucleus. – 2015. – No. 46. – P. 1770-1830.
- 3 Neutron Capture Therapy: Principles and Applications. / W. Sauerwein [at al]: Springer-Verlag. – 2012. – 543 p.
- 4 Таскаев С.Ю. Бор-нейтронозахватная терапия. / С.Ю. Таскаев, В.В. Каныгин – Новосибирск: изд-во СО РАН, – 2016. – 216 с.
- 5 Koshkarev A.M. Development of automation system of the ion source. / A.M Koshkarev [at al] // Proceedings of RUPAC. – 2014. – No. 06-10. – P. 380-382.
- 6 Koshkarev A.M. Development and Implementation of the Automation System of the Ion Source for BNCT. / A.M Koshkarev [at al] // Proceedings of the XXV Russian Particle Accelerator Conference. – 2016. – P. 21-25.

ПРИЛОЖЕНИЕ А

Пример конфигурационного файла

```
<?xml version="1.0" encoding="utf-8" ?>
<Root>
  <Application>
    <Base
      ShowFullFormLog="true"
      ExcelLogFiles="D:\Users\kent_brockman4\soft\tandem\ok_cfg_no_calibr\Log\BENS Control Program"
    />
  </Application>

  <Modules>
    <Modbus>
      <Pet7215>
        <CooledDiafragm displayName="Охлаждаемая диафрагма и после магнита" IP="192.168.0.152"/>
      </Pet7215>
      <Adam6024>
      </Adam6024>
      <Adam6066>
        <Reley displayName="Реле в боксе" IP="192.168.0.103"/>
      </Adam6066>
    </Modbus>
    <COM>
      <Pyrometer displayName="пирометр" port="COM7" baudRate="115200" dataBits="8"/>
    </COM>
    <SCPI>
      <PSU6200>
        <HminusMagnet displayName="H- питание магнита" IP="192.168.0.15"/>
      </PSU6200>
    </SCPI>
  </Modules>

  <Devices>
    <ChillingDiaphragms displayName="Охлаждаемые диафрагмы" voiceEnabled="true">
      <BeforeMagnet displayName="Диафрагма до магнита" vectorGraphColor="0x0000FF"
        vectorGraphDisplayName="до магнита" onVectorLengthVeryBigMessageName ="до магнита"
        soundFileName="cooledDiaphragm01">
        <Up displayName="Вверх" module="CooledDiafragm" channel="0"/>
        <Right displayName="Право" module="CooledDiafragm" channel="1"/>
        <Down displayName="Низ" module="CooledDiafragm" channel="2"/>
        <Left displayName="С" module="CooledDiafragm" channel="3"/>
        <MaxTemperature displayName="Мак до магнита" color="0x0000FF"/>
      </BeforeMagnet>

      <AfterMagnet displayName="диафрагма после магнита" vectorGraphColor="0x00FF00"
        vectorGraphDisplayName="после магнита" onVectorLengthVeryBigMessageName ="после магнита"
        soundFileName="temperatureAfterMagnet">
        <A displayName="А, H-" module="CooledDiafragm" channel="4"/>
        <B displayName="B" module="CooledDiafragm" channel="5"/>
        <C displayName="C" module="CooledDiafragm" channel="6"/>
        <MaxTemperature displayName="Мак после магнита" color="0x00FF00"/>
      </AfterMagnet>
    </ChillingDiaphragms>

    <StrippingTarget>
      <TemperatureIn displayName="Входная температура" module="StrippingTargetTemperature"
        channel="0"/>
      <TemperatureOut displayName="Выходная температура" module="StrippingTargetTemperature"
        channel="1"/>
      <FlowMeter>
        <FlowAdc displayName="поток АЦП" module="StrippingTargetFlowAdc" channel="5"/>
        <TemperatureAdc displayName="Темпеатура АЦП" module="StrippingTargetFlowAdc" channel="4"/>
        <PressureAdc displayName="Давление АЦП" module="StrippingTargetFlowAdc" channel="3"/>

        <FlowVirtual displayName="поток (LPM) virtual" color="0x000000"/>
        <TemperatureVirtual displayName="Темпеатура virtual" color="0x000000"/>
        <PressureVirtual displayName="давление virtual" color="0x000000"/>
      </FlowMeter>
      <Power displayName="Мощность" color="0x0000FF"/>
    </StrippingTarget>

    <Rebutator displayName="Ребуталка">
      <Power220Get displayName="220 в коробке (get)" module="Reley" channel="0"/>
      <Power220Set displayName="220 в коробке (set)" module="Reley" channel="0"/>
      <MediaConverterOnRackSet displayName="Медиаконвертер у ИП (set)" module="Reley" channel="1"/>
      <ArgonGasControllerSet displayName="Напуск газа (set)" module="Reley" channel="2"/>

      <CAN displayName="CAN" IP="192.168.0.250"/>
      <ArgonGasController displayName="Напуск аргона" IP="192.168.0.90"/>
    </Rebutator>

    <Buzzer300V displayName="Пищалка" defaultState="true">
  </Devices>
</Root>
```

```

    <PowerGet displayName="питание пищалки (get) Сырое" module="Reley" channel="1"/>
    <PowerGetReal displayName="питание пищалки (get) Реальное"/>
    <PowerSet displayName="питание пищалки (set)" module="Reley" channel="3"/>
</Buzzer300v>

<Gates displayName="Шиберы">
    <TMNAcceleratorInSet displayName="шибер ТМН на входе в ускоритель" module="Reley"
channel="4"/>
</Gates>

<Pyrometer displayName="пирометр">
    <Pyrometer displayName="пирометр" color="0xFF0000" module="Pyrometer"/>
</Pyrometer>

<HMinus displayName="H-">
    <RotateMagnet displayName="Поворотный магнит">
        <Voltage displayName="U" module="HminusMagnet"/>
        <Current displayName="I" module="HminusMagnet"/>
    </RotateMagnet>
</HMinus>

<TestDevice displayName="Тестовое устройство">
    <TestValue displayName="Тестовое значение"/>
</TestDevice>
</Devices>

<Utilites>
    <FluenceCounter displayName="Расчет флюенса"
dataFolder="D:\Users\kent_brockman4\soft\tandem\ok_cfg_no_calibr"/>
</Utilites>
</Root>

```



```

        tlvDeviceChannelsValues.CellEditActivation =
ObjectListView.CellEditActivateMode.DoubleClick;
    }
    tlvDeviceChannelsValues.Refresh();
}
private void ConfigureTreeViewGetters()
{
    tlvDeviceChannelsValues.CanExpandGetter = delegate(object x)
    {
        return (x is Device.Parent);
    };
    tlvDeviceChannelsValues.ChildrenGetter = delegate(object x)
    {
        var device = (Device.Parent)x;

        var result = new List<object>();
        result.AddRange(device.channels);
        result.AddRange(device.childDevices);

        return result;
    };
    tlvDeviceChannelsValues.CheckStateGetter = delegate(object x)
    {
        if (x == null)
            return CheckState.Unchecked;

        if (x is Device.Parent)
            return ((Device.Parent)x).Connected;

        if (x is Channel.Real.Parent<double>)
            return ((Channel.Real.Parent<double>)x).Connected ? CheckState.Checked :
CheckState.Unchecked;

        if (x is Channel.Real.Parent<bool>)
            return ((Channel.Real.Parent<bool>)x).Connected ? CheckState.Checked :
CheckState.Unchecked;

        if (x is Channel.Virtual.Parent<double>)
            return ((Channel.Virtual.Parent<double>)x).Connected ? CheckState.Checked :
CheckState.Unchecked;

        if (x is Channel.Virtual.Parent<bool>)
            return ((Channel.Virtual.Parent<bool>)x).Connected ? CheckState.Checked :
CheckState.Unchecked;

        throw new NotImplementedException();
    };
    olvColumnDeviceChanel.AspectGetter = delegate(object x)
    {
        if (x is Device.Parent)
            return ((Device.Parent)x).ToString();

        if (x is Channel.Parent)
            return ((Channel.Parent)x).DisplayName;

        throw new NotImplementedException();
    };
    olvColumnModule.AspectGetter = delegate(object x)
    {
        if (x is Channel.Real.Parent<bool>)
            return ((Channel.Real.Parent<bool>)x).module.ToString();

        if (x is Channel.Real.Parent<double>)
            return ((Channel.Real.Parent<double>)x).module.ToString();

        if (x is Channel.Virtual.ReadOnly<double>)
            return "Вычисляемое значение (Read only)";

        if (x is Channel.Virtual.ReadOnly<bool>)
            return "Вычисляемое значение (Read only)";

        if (x is Channel.Virtual.Readwrite<double>)
            return "Вычисляемое значение (Read/write)";

        if (x is Channel.Virtual.Readwrite<bool>)
            return "Вычисляемое значение (Read/write)";

        return "";
    };
    olvColumnValue.AspectGetter = delegate(object x)
    {
        if (x is Channel.ParentGeneric<bool>)

```

```

        return ((Channel.ParentGeneric<bool>)x).Value().ToString();
    if (x is Channel.ParentGeneric<double>)
        return ((Channel.ParentGeneric<double>)x).Value().ToString("0.0");
    if (x is Device.Parent)
        return "";
    throw new NotImplementedException();
};

olvColumnValue.AspectPutter = delegate(object channel, object newValue)
{
    try
    {
        if (channel is Channel.Interfaces.Iwrite<double> channelDouble)
        {
            var value = double.Parse(newValue.ToString());
            channelDouble.Value(value);
            return;
        }

        if (channel is Channel.Interfaces.Iwrite<bool> channelBool)
        {
            var value = bool.Parse(newValue.ToString());
            channelBool.Value(value);
            return;
        }

        ShowMessageBoxInTreeView("Этот канал настроен только на чтение");
    }
    catch (Exception ex)
    {
        ShowMessageBoxInTreeView(ex.Message);
    }
};

private void ShowMessageBoxInTreeView(string message)
{
    this.BeginInvoke((MethodInvoker)delegate()
    {
        MessageBox.Show(this, message, "Error", MessageBoxButtons.OK,
        MessageBoxIcon.Exclamation);
    });
}

private void ConfigureExcelLogs()
{
    timeLog = new TimeLog(true, Channel.ParentStatic.GetAllChannels().Select(x =>
x.LogName).ToArray(), "Log")
    {
        remotedirectory = new DirectoryInfo(ApplicationConfig.Parent.ExcelLogFiles)
    };
    timeLog.OnBGWColmletedWithException += FileLog_OnBGWColmletedWithException;
}

void FileLog_OnBGWColmletedWithException(object sender, Exception ex)
{
    var message = String.Format("Во время записи логов произошла ошибка!\n ЛОГИ ПЕРЕСТАЛИ
ЗАПИСЫВАТЬСЯ\n\n'{0}'", ex.Message);
    AddToExcelLogBuffer(message);
    throw new IOException(message, ex);
}

private void StartUpdating()
{
    timer100.Enabled = true;
    timer500.Enabled = true;
    timer1000.Enabled = true;

    Module.Parent.AllModulesUpdating = true;

    CommitFileLog();
    timeLog.Push();
}

private void ReadConfig()
{
    ApplicationConfig.Parent.Init("Application", "Base");

    ConfigureModules();
    ConfigureDevicesEvents();
    ConfigureDevicesControls();
}

```



```

        ConfigureChannels();
        ConfigureUtilites();
    }
    private void ConfigureUtilites()
    {
        ConfigureFluenceCounter();
    }
    void FormLog_OnPushing(string[] buffer)
    {
        foreach (var e in buffer)
        {
            AddToExcelLogBuffer(e);
        }
    }
    private void ConfigureDevicesControls()
    {
        ConfigurePyrometer();
        ConfigureTemperatureProfilometer();
#if ENABLE_HMINUS
        ConfigureHMinus();
#endif
    }
    private void ConfigureTemperatureProfilometer()
    {
        vectorGraph = new
Device.ChillingDiaphragms.DiaphragmsType.VectorGraph(graphVectorGraph);
        vectorGraph.HideLicenceBar(pnlVectorGraph);
    }
    private void ConfigurePyrometer()
    {
        pyrometerTemperature = new Round(graphPyrometerTemperature,
Device.ParentStatic.pyrometer.displayName, 0, 350, 370, 400);
        pyrometerTemperature.HideLicenceBar(pyrometerTemperature_PNL);
        pyrometerTemperature.SetScale(50, 10, 0);
    }
    private void ConfigureHMinus()
    {
        barRWVoltage.Text = Device.ParentStatic.HMinus.RotateMagnet.Voltage.DisplayName;
        barRWCurrent.Text = Device.ParentStatic.HMinus.RotateMagnet.Current.DisplayName;
        gbRotateMagnet.Text = Device.ParentStatic.HMinus.RotateMagnet.displayName;
    }
    #region FluenceCounter
    private void ConfigureFluenceCounter()
    {
        Utilites.FluenceCounter.Parent.Init("Utilites", "FluenceCounter");
        Utilites.FluenceCounter.Parent.OnProgressChanged += Engine_OnProgressChanged;
        Utilites.FluenceCounter.Parent.OnBGWColmletedWithException +=
Engine_OnBGWColmletedWithException;
        Utilites.FluenceCounter.Parent.OnFluenceCalculated += Engine_OnFluenceCalculated;
        textPathToLog.Text = Utilites.FluenceCounter.Parent.pathToLog;
    }
    void Engine_OnFluenceCalculated(double fluence)
    {
        var fluencemAh = String.Format("{0:0.00} mAh", fluence / 3600000);
        var fluencemAm = String.Format("{0:0000.00} mA", fluence / 60000);

        FormLog.Commit(String.Format("Флюенс расчитан: {0}, {1}", fluencemAh, fluencemAm));
        lblFluenceValuemAh.Text = fluencemAh;
        lblFluenceValuemAm.Text = fluencemAm;
    }
    void Engine_OnBGWColmletedWithException(Exception ex)
    {
        FormLog.Commit(String.Format("FluenceCounter calculation error '{0}'", ex),
FormLog.Type.Error);
    }
    void Engine_OnProgressChanged(object sender, ProgressChangedEventArgs e)
    {
        pbarFluence.Value = e.ProgressPercentage;
    }
    #endregion
    private void ConfigureChannels()
    {
        Channel.ParentStatic.Init();
    }

```

```

private void ConfigureDevicesEvents()
{
    Device.ParentStatic.Init();

    Device.ParentStatic.chillingDiaphragms.afterMagnet.OnVectorLengthVeryBig +=
OnDiaphragmVectorLengthVeryBig;
    Device.ParentStatic.chillingDiaphragms.beforeMagnet.OnVectorLengthVeryBig +=
OnDiaphragmVectorLengthVeryBig;

    Device.ParentStatic.rebutator.OnProgressChanged += Rebutator_OnProgressChanged;
    Device.ParentStatic.rebutator.OnShowWait += Rebutator_OnShowWait;
    Device.ParentStatic.rebutator.OnProcessingFinished += Rebutator_OnProcessingFinished;
    Device.ParentStatic.rebutator.OnBGWColmletedWithException +=
Rebutator_OnBGWColmletedWithException;

    foreach (var device in Device.ParentStatic.list)
    {
        device.OnConnectionEstablished += Device_OnConnectionEstablished;
        device.OnConnectionFailed += Device_OnConnectionFailed;
        device.OnConnectionLost += Device_OnConnectionLost;

        foreach (var childDevice in device.childDevices)
        {
            childDevice.OnConnectionEstablished += Device_OnConnectionEstablished;
            childDevice.OnConnectionFailed += Device_OnConnectionFailed;
            childDevice.OnConnectionLost += Device_OnConnectionLost;
        }
    }
}

void Device_OnConnectionEstablished(Device.Parent device)
{
    var message = String.Format("Устройство '{0}' подключено", device.ToString());
    FormLog.Commit(message, FormLog.Type.Good);
}

void Device_OnConnectionFailed(Device.Parent device, object channel, Exception ex)
{
    var shortMessage = String.Format("Подключение к устройству '{0}' в канале '{1}'
возвратило ошибку", device.ToString(), channel.ToString());
    var fullmessage = String.Format("{0} '{1}'", shortMessage, ex.Message);

    OnConnection_ProcessLogging(shortMessage, fullmessage);
}

void Device_OnConnectionLost(Device.Parent device, object channel, Exception ex)
{
    var shortMessage = String.Format("Связь с устройством '{0}' в канале '{1}' потеряна",
device.ToString(), channel.ToString());
    var fullmessage = String.Format("{0} из-за ошибки '{1}'", shortMessage, ex.Message);

    OnConnection_ProcessLogging(shortMessage, fullmessage);
}

private void OnConnection_ProcessLogging(string shortMessage, string fullmessage)
{
    if (ApplicationConfig.Parent.ShowFullFormLog)
    {
        FormLog.Commit(fullmessage, FormLog.Type.Error, true);
    }
    else
    {
        AddToExcelLogBuffer(fullmessage);
        FormLog.Commit(shortMessage, FormLog.Type.Error, false);
    }
}

private void AddToExcelLogBuffer(string message)
{
    lock (excelLogBuffer)
    {
        excelLogBuffer.Append(message + "; ");
    }
}

private void Rebutator_OnProcessingFinished(object sender, RunworkerCompletedEventArgs e)
{
    btnReboot.Text = "Ребути ребутай";

    if (e.Cancelled)
    {
        FormLog.EditLast("Перезагрузка отменена! Включаю модули..", FormLog.Type.Warning);
        return;
    }
}

```

```

        if (!(bool)e.Result)
        {
            FormLog.EditLast("Перезагрузка закончена, но не все модули включились!",
FormLog.Типе.Warning);
            return;
        }
        FormLog.EditLast("Перезагрузка прошла успешно", FormLog.Типе.Good);
    }

    void Rebutator_OnShowwait(object sender, string message)
    {
        FormLog.EditLast(message);
    }

    void Rebutator_OnProgressChanged(object sender, ProgressChangedEventArgs e)
    {
        pbarRebutator.Value = e.ProgressPercentage;
    }

    void Rebutator_OnBGWColmletedWithException(object sender, Exception ex)
    {
        FormLog.Commit(String.Format("В процессе ребутания возникла ошибка '{0}'",
ex.ToString()), FormLog.Типе.Error);
        throw ex;
    }

    void OnDiaphragmVectorLengthVeryBig(Device.ChillingDiaphragms.DiaphragmsType.PositionInfo
tmpDirection, Device.ChillingDiaphragms.DiaphragmsType.PositionInfo tmpSource)
    {
        var message = String.Format("Пучек {0} на {1}", tmpDirection.comment,
tmpSource.comment);
    }

    private void ConfigureModules()
    {
        Module.Parent.Init(1stModules);

        foreach (var e in Module.Parent.modules)
        {
            e.OnConnectionEstablished += Module_OnConnectionEstablished;
            e.OnConnectionFailed += Module_OnConnectionFailed;
            e.OnConnectionLost += Module_OnConnectionLost;
            e.OnConnectionRestored += Module_OnConnectionRestored;
            e.OnBGWColmletedWithException += Module_OnBGWColmletedWithException;
        }
    }

    private void ConfigureForm()
    {
        InitializeComponent();
#if !ENABLE_HMINUS
        tabControl1.TabPages.Remove(tabPage6);
#endif
        this.StartPosition = FormStartPosition.Manual;
        this.Location = new Point(50, 10);
#if DEBUG
        this.Text = AppVersion.PrintVersion("DEBUG Budker Epithermal Neutron Source -
ControlProgram");
#else
        this.Text = AppVersion.PrintVersion("Budker Epithermal Neutron Source -
ControlProgram");
#endif

        excelLogBuffer = new StringBuilder();
        AddToExcelLogBuffer(this.Text);
        FormLog.Init(1stLog);
    }

    private void ConfigureRealTimeGraph()
    {
        var seriesInfo = new List<SeriesInfo>();
        foreach (var e in Channel.ParentStatic.GetGraphChannels())
        {
            seriesInfo.Add(new SeriesInfo(e.DisplayName, e.Color));
        }
        realTimeGraph = new RealTime(graphRealTime, hsbRealTime, "Temperature", updateInterval,
seriesInfo.ToArray());
    }

    void Module_OnConnectionEstablished(KB4.Modulewizard.Module module)
    {
        if (showModuleconnected)
        {

```

```

        var message = String.Format("Подключение к модулю '{0}' успешно",
module.ToString());
        AddToExcelLogBuffer(message);
    }
}

void Module_OnConnectionFailed(КВ4.ModuleWizard.Module module, Exception ex)
{
    var message = String.Format("Подключение к модулю '{0}' возвратило ошибку '{1}'",
module.ToString(), ex.Message);
    AddToExcelLogBuffer(message);
}

void Module_OnConnectionLost(КВ4.ModuleWizard.Module module, Exception ex)
{
    var message = String.Format("Связь с модулем '{0}' потеряна из-за ошибки '{1}'",
module.DisplayName, ex.Message);
    FormLog.Commit(message, FormLog.Type.Warning);
}

void Module_OnConnectionRestored(КВ4.ModuleWizard.Module module)
{
    var message = String.Format("Связь с модулем '{0}' восстановлена",
module.DisplayName);
    FormLog.Commit(message, FormLog.Type.Good);
}

void Module_OnBGWColmletedWithException(КВ4.ModuleWizard.Module sender, Exception ex)
{
    throw ex;
}

private void Timer100_Tick(object sender, EventArgs e)
{
    FormLog.Push();
}

private void CommitFileLog()
{
    var values = Channel.ParentStatic.GetAllChannels().Select(x => x.ValueStr).ToArray();
    timeLog.Add(values);
    lock (excelLogBuffer)
    {
        timeLog.Add(excelLogBuffer.ToString());
        excelLogBuffer.Clear();
    }
    timeLog.Commit();
}

private void UpdateControls()
{
    UpdateGraphs();
    UpdateModuleLists();

    UpdateFluenceCounterStopDate();
    UpdateRebutator();
    UpdateBuzzer300V();
    UpdateGates();
#if ENABLE_HMINUS
    UpdateHMinus();
#endif
    UpdateDiploma();

    UpdateAllDeviceStatus();
}

private void UpdateDiploma()
{
    ledControl1.valueCheckState = Device.ParentStatic.Diploma.Connected;
    ledControl2.valueBool = Device.ParentStatic.Diploma.Led.Value();

    if (ledControl2.valueBool)
    {
        button1.Text = "Выключить";
    }
    else
    {
        button1.Text = "Включить";
    }

    button1.Enabled = true;

    ledControl3.valueCheckState = Device.ParentStatic.Diploma.Connected;
    ledControl4.valueBool = Device.ParentStatic.Diploma.Button1.Value();
    ledControl5.valueBool = !Device.ParentStatic.Diploma.Button2.Value();
}

```

```

        ledControl6.valueBool = Device.ParentStatic.Diploma.And.Value();
    }

    private void UpdateHMinus()
    {
        barRWVoltage.Value = Device.ParentStatic.HMinus.RotateMagnet.Voltage.Value();
        barRWCurrent.Value = Device.ParentStatic.HMinus.RotateMagnet.Current.Value();
        ledHMinusRotateMagnetConnected.valueCheckState =
Device.ParentStatic.HMinus.RotateMagnet.Connected;
    }

    private void UpdateAllDeviceStatus()
    {
        var allDevicesConnections = Device.ParentStatic.list.Select(x =>
x.Connected).ToArray();
        var connected = Device.Parent.GetConnectionState(allDevicesConnections);
        ledAllDeviceStatus.valueCheckState = connected;
    }

    private void UpdateBuzzer300V()
    {
        ledBuzzer300VDeviceConnected.valueCheckState =
Device.ParentStatic.buzzer300V.Connected;
        ledBuzzer300VStatus.valueBool = Device.ParentStatic.buzzer300V.PowerGetReal.Value();
        if (ledBuzzer300VStatus.valueBool)
        {
            btnBuzzerPower.Text = "Выключить";
        }
        else
        {
            btnBuzzerPower.Text = "Включить";
        }

        btnBuzzerPower.Enabled = true;
    }

    private void UpdateGates()
    {
        ledGateTMNAcceleratorInDeviceConnected.valueCheckState =
Device.ParentStatic.gates.Connected;
        ledGateTMNAcceleratorInStatus.valueBool =
Device.ParentStatic.gates.TMNAcceleratorIn.Value();
        if (ledGateTMNAcceleratorInStatus.valueBool)
        {
            btnGateTMNAcceleratorIn.Text = "Закрыть";
        }
        else
        {
            btnGateTMNAcceleratorIn.Text = "Открыть";
        }

        btnGateTMNAcceleratorIn.Enabled = true;
    }

    private void UpdateRebutator()
    {
        ledRebutatorDeviceConnected.valueCheckState = Device.ParentStatic.rebutator.Connected;
        btnReboot.Enabled = Device.ParentStatic.rebutator.Connected == CheckState.Checked;
        ledRebutatorCanStatus.valueBool = Device.ParentStatic.rebutator.CANStatus.Value();
        ledRebutatorArgonGasControllerStatus.valueBool =
Device.ParentStatic.rebutator.ArgonGasControllerStatus.Value();
    }

    private void UpdateFluenceCounterStopDate()
    {
        if (cbxCurrentTime.Checked)
            dateStopDate.Value = DateTime.Now;
    }

    private void UpdateModuleLists()
    {
        Module.Parent.UpdateStatus();
    }

    private void UpdateGraphs()
    {
        UpdateRealTimeGraph();
        UpdateRoundGraph();
        UpdateVectorGraph();
    }

    private void UpdateVectorGraph()
    {
        vectorGraph.Draw();
    }

    private void UpdateRoundGraph()
    {

```

```

    pyrometerTemperature.Draw(Device.ParentStatic.pyrometer.module.Value());
}
private void UpdateRealTimeGraph()
{
    realTimeGraph.Add(Channel.ParentStatic.GetGraphChannels().Select(x =>
x.Value()).ToArray());
    realTimeGraph.Update();
}
private void MainForm_FormClosed(object sender, FormClosedEventArgs e)
{
    timeLog.Dispose();
    Utilites.FluenceCounter.Parent.Dispose();

    Module.Parent.AllModulesUpdating = false;
}
private void Date_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        e.Handled = true;
        e.SuppressKeyPress = true;
        FormLog.Commit("Начинаю расчет флюенса");
        Utilites.FluenceCounter.Parent.StartCalc(dateStartDate.Value, dateStopDate.Value);
    }
}
private void CbxCurrentTime_CheckedChanged(object sender, EventArgs e)
{
    dateStopDate.Enabled = !cbxCurrentTime.Checked;
}
private void BtnCalcFluence_Click(object sender, EventArgs e)
{
    FormLog.Commit("Начинаю расчет флюенса");
    Utilites.FluenceCounter.Parent.StartCalc(dateStartDate.Value, dateStopDate.Value);
}
private void BtnReboot_Click(object sender, EventArgs e)
{
    if (Device.ParentStatic.rebutator.IsBusy)
    {
        Device.ParentStatic.rebutator.IsBusy = false;
        btnReboot.Text = "Ребути ребутай";
    }
    else
    {
        FormLog.Commit("Начато ребутание");
        FormLog.Commit("Ожидайте");
        Device.ParentStatic.rebutator.IsBusy = true;
        btnReboot.Text = "Отмена";
    }
}
private void BtnBuzzerPower_Click(object sender, EventArgs e)
{
    btnBuzzerPower.Enabled = false;
    Device.ParentStatic.buzzer300V.PowerSet.Value(!ledBuzzer300VStatus.valueBool);
}
private void barRWvoltage_ValueChanged(object sender, double value)
{
    Device.ParentStatic.HMinus.RotateMagnet.Voltage.Value(value);
}
private void btnRotateMagnetStop_Click(object sender, EventArgs e)
{
    Device.ParentStatic.HMinus.RotateMagnet.Module.Abort();
}
private void barRWcurrent_ValueChanged(object sender, double value)
{
    Device.ParentStatic.HMinus.RotateMagnet.Current.Value(value);
}
private void BtnOpenLogFolder_Click(object sender, EventArgs e)
{
    Process.Start(timeLog.remoteDirectory.FullName);
}
private void BtnOpenExeFolder_Click(object sender, EventArgs e)
{
    Process.Start(new
FileInfo(System.Reflection.Assembly.GetEntryAssembly().Location).Directory.FullName);
}

```

```

    }
    private void BtnOpenConfigFile_Click(object sender, EventArgs e)
    {
        Process.Start("notepad.exe", Path.Combine(AppDomain.CurrentDomain.BaseDirectory,
"Config.xml"));
    }

    private void Timer500_Tick(object sender, EventArgs e)
    {
        UpdateControls();
        tlvDeviceChannelsValues.Refresh();
    }

    private void Timer1000_Tick(object sender, EventArgs e)
    {
        if (!showModuleConnected)
        {
            if (showModuleConnectedTimeoutSec > 0)
                showModuleConnectedTimeoutSec--;
            else
                showModuleConnected = true;
        }
        CommitFileLog();
    }

    private void BtnGateTMNAcceleratorIn_Click(object sender, EventArgs e)
    {
        btnGateTMNAcceleratorIn.Enabled = false;
Device.ParentStatic.gates.TMNAcceleratorIn.Value(!ledGateTMNAcceleratorInStatus.valueBool);
    }

    private void button1_Click(object sender, EventArgs e)
    {
        button1.Enabled = false;
        Device.ParentStatic.Diploma.Led.Value(!ledControl2.valueBool);
    }
}
}
}

```

ПРИЛОЖЕНИЕ В

Листинг программного файла Module.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using EasyModbus;

namespace KB4.Modulewizard
{
    public delegate void ConnectionSuccess(Module module);
    public delegate void ConnectionError(Module module, Exception ex);

    public abstract class Module
    {
        public event ConnectionSuccess OnConnectionEstablished;
        public event ConnectionSuccess OnConnectionRestored;
        public event ConnectionError OnConnectionFailed;
        public event ConnectionError OnConnectionLost;
        public event ConnectionError OnBGWColmletedWithException;

        protected abstract void GetDataAsync();
        protected abstract void SetDataAsync();

        protected abstract void GetInitalState();

        public abstract void Connect();
        public abstract bool Connected { get; }

        public string Name { get; set; }
        public string DisplayName { get; set; }

        public bool AsyncDataReady { get; private set; }

        public const int updateIntervalMS = 500;
        public const int connectTimeoutMS = 3000;
        public const int ReadwriteTimeoutMS = 1000;

        BackgroundWorker backgroundWorker { get; set; }
        bool connectionRestored = false;

        public Module(string name, string displayName)
        {
            this.Name = name;
            this.DisplayName = displayName;

            AsyncDataReady = false;
            backgroundWorker = new BackgroundWorker();
            backgroundWorker.DoWork += bgw_DoWork;
            backgroundWorker.RunWorkerCompleted += backgroundWorker_RunworkerCompleted;
        }

        void backgroundWorker_RunworkerCompleted(object sender, RunworkerCompletedEventArgs e)
        {
            if (e.Error != null)
            {
                if (OnBGWColmletedWithException != null)
                    OnBGWColmletedWithException(this, e.Error);
                else
                    throw e.Error;
            }
        }

        bool riseConnectionFailed = true;
        bool riseConnectionLost = true;

        bool _updating;
        public bool Updating
        {
            get
            {
                return _updating;
            }
            set
            {
                if (value)
                    backgroundWorker.RunworkerAsync();
                _updating = value;
            }
        }
    }
}
```



```

void bgw_DoWork(object sender, DoWorkEventArgs e)
{
    try
    {
        Thread.CurrentThread.Name = this.ToString();
        PrepareThread();
        while (_updating)
        {
            AsyncDataReady = false;
            if (Connected)
            {
                TrySetGetData();
            }
            else
            {
                TryConnect();
            }
            Thread.Sleep(updateIntervalMS);
        }
    }
    catch (Exception ex)
    {
        throw new ApplicationException(String.Format("Ошибка в потоке '{0}'",
this.ToString()), ex);
    }
}

protected virtual void PrepareThread() { }

private void TrySetGetData()
{
    try
    {
        SetDataAsync();
        GetDataAsync();
        AsyncDataReady = true;
        if (connectionRestored)
        {
            connectionRestored = false;
            BaseOnConnectionRestored();
        }
    }
    catch (Exception ex)
    {
        BaseOnConnectionLost(ex);
    }
}

private void TryConnect()
{
    try
    {
        Connect();
        GetInitialState();
        BaseOnConnectionEstablished();
    }
    catch (Exception ex)
    {
        BaseOnConnectionFailed(ex);
    }
}

void BaseOnConnectionEstablished()
{
    riseConnectionFailed = true;
    if (OnConnectionEstablished == null)
        throw new MissingMethodException("Метод OnConnectionEstablished не реализован!");
    OnConnectionEstablished(this);
}

void BaseOnConnectionFailed(Exception ex)
{
    if (riseConnectionFailed)
    {
        riseConnectionFailed = false;
        if (OnConnectionFailed == null)
            throw new MissingMethodException("Метод OnConnectionFailed не реализован!",
ex);
        OnConnectionFailed(this, ex);
    }
}

void BaseOnConnectionLost(Exception ex)
{
    if (riseConnectionLost)
    {

```

```

        riseConnectionLost = false;
        connectionRestored = true;
        if (OnConnectionLost == null)
            throw new MissingMethodException("Метод OnConnectionLost не реализован!", ex);
        OnConnectionLost(this, ex);
    }
}

void BaseOnConnectionRestored()
{
    riseConnectionLost = true;
    if (OnConnectionRestored == null)
        throw new MissingMethodException("Метод OnConnectionRestored не реализован!");
    OnConnectionRestored(this);
}

public override string ToString()
{
    return String.Format("{0}: {1}", base.ToString().Split(new char[] { '.' }).Last(),
        DisplayName);
}
}
}
}

```

ПРИЛОЖЕНИЕ Г

Листинг программного файла Adam6066.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace KB4.Modulewizard.Modbus.Adam
{
    public class Adam6066 : Parent, Interfaces.IDi, Interfaces.IRelay
    {
        public Interfaces.DigitalInfo InfoDi { get; private set; }
        public bool[] DiGetValueAsync { get; set; }

        public Interfaces.DigitalInfo InfoRelay { get; private set; }
        public bool[] RelayGetValueAsync { get; set; }
        public bool[] RelaySetValueAsync { get; set; }
        public bool RelaySetValue { get; set; }

        public Adam6066(string name, string displayName, string IP)
            : base(name, displayName, IP)
        {
            InfoDi = new Interfaces.DigitalInfo(0, 6);
            InfoRelay = new Interfaces.DigitalInfo(16, 6);

            DiGetValueAsync = new bool[InfoDi.length];
            RelayGetValueAsync = new bool[InfoRelay.length];
            RelaySetValueAsync = new bool[InfoRelay.length];
        }

        #region Di
        /// <summary>
        /// Getter for DI
        /// </summary>
        /// <param name="channel">Channel index. Starting from 0</param>
        /// <returns>DI value</returns>
        public bool Di(int channel)
        {
            Engine.InputCheck(InfoDi, channel);
            return DiGetValueAsync[channel];
        }

        /// <summary>
        /// Getter for DI
        /// </summary>
        /// <returns>All DI values</returns>
        public bool[] Di()
        {
            return DiGetValueAsync;
        }
        #endregion

        #region Relay
        /// <summary>
        /// Getter for Relay
        /// </summary>
        /// <param name="channel">Channel index. Starting from 0</param>
        /// <returns>Relay value</returns>
        public bool Relay(int channel)
        {
            Engine.InputCheck(InfoDi, channel);
            return RelayGetValueAsync[channel];
        }

        /// <summary>
        /// Getter for Relay
        /// </summary>
        /// <returns>All Relay values</returns>
        public bool[] Relay()
        {
            return RelayGetValueAsync;
        }

        /// <summary>
        /// Setter for Relay
        /// </summary>
        /// <param name="channel">Channel index. Starting from 0</param>
        /// <param name="value">New Relay value</param>
        public void Relay(int channel, bool value)
        {
            Engine.InputCheck(InfoDi, channel);

```

```

        RelaySetValueAsync[channel] = value;
        RelaySetValue = true;
    }

    /// <summary>
    /// Setter for Relay
    /// </summary>
    /// <param name="start">Channel start index. Starting from 0</param>
    /// <param name="value">New Relay values</param>
    public void Relay(int start, bool[] value)
    {
        Engine.InputCheck(InfoDi, start, value.Length);
        System.Array.Copy(value, 0, RelaySetValueAsync, start, value.Length);
        RelaySetValue = true;
    }
    #endregion

    protected override void GetDataAsync()
    {
        DiGetValueAsync = Engine.Di(this, 0, InfoDi.length);
        RelayGetValueAsync = Engine.Relay(this, 0, InfoRelay.length);
    }

    protected override void SetDataAsync()
    {
        if (RelaySetValue)
        {
            RelaySetValue = false;
            Engine.Relay(this, 0, RelaySetValueAsync);
        }
    }

    protected override void GetInitialState()
    {
        var currentValues = Engine.Relay(this, 0, InfoRelay.length);
        for (int i=0; i < InfoRelay.length; i++)
        {
            RelaySetValueAsync[i] |= currentValues[i];
        }
    }
}
}
}

```

Цветная печать

Авто

7,7,9,9,10,12,13,13,14,14,15,15,18,20,22,26,34,35,36,37,38,39,43,45,45,46,47,

48

На печать