TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа – Инженерная школа информационных технологий и робототехники
Направление подготовки – 15.04.06 Мехатроника и робототехника
ООП Управление робототехническими комплексами и мехатронными системами
Отделение школы (НОЦ) – Отделение автоматизации и робототехники

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРАНТА

Тема работы
Разработка программно-аппаратного комплекса для 4-х тросовой системы обезвешивания

УДК 681.5:624.071.2

Обучающийся

Группа	ФИО	Подпись	Дата
8ЕМ11	Иванов Егор Андреевич		

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Заведующий кафедрой – руководитель отделения на правах кафедры ОАР ИШИТР	Филипас А.А.	к.т.н., доцент		

Консультант

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Старший преподаватель ОАР ИШИТР	Беляев А.С.	–		

КОНСУЛЬТАНТЫ ПО РАЗДЕЛАМ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОСГН ШБИП	Былкова Т.В.	канд. экон. наук, доцент		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор ООД ШБИП	Федорчук Ю.М.	д-р техн. наук, профессор		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОАР ИШИТР	Кузьминская Е.В.	к.т.н., доцент		

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
Заведующий кафедрой – руководитель отделения на правах кафедры ОАР ИШИТР	Филипас А.А.	к.т.н., доцент		

Томск – 2023 г.

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОСВОЕНИЯ ООП

Код компетенции	Наименование компетенции
Универсальные компетенции	
УК(У)-1	Способен осуществлять критический анализ проблемных ситуаций на основе системного подхода, вырабатывать стратегию действий.
УК(У)-2	Способен управлять проектом на всех этапах его жизненного цикла
УК(У)-3	Способен организовывать и руководить работой команды, вырабатывая командную стратегию для достижения поставленной цели.
УК(У)-4	Способен применять современные коммуникативные технологии, в том числе на иностранном(-ых) языке(-ах), для академического и профессионального взаимодействия.
УК(У)-5	Способен анализировать и учитывать разнообразие культур в процессе межкультурного взаимодействия.
УК(У)-6	Способен определять и реализовывать приоритеты собственной деятельности и способы ее совершенствования на основе самооценки.
Общепрофессиональные компетенции	
ОПК(У)-1	Способен применять естественнонаучные и общеинженерные знания, методы математического анализа и моделирования в профессиональной деятельности.
ОПК(У)-2	Способен применять основные методы, способы и средства получения, хранения, переработки информации в области машиностроения.
ОПК(У)-3	Способен осуществлять профессиональную деятельность с учетом экономических, экологических, социальных и других ограничений на всех этапах жизненного уровня.
ОПК(У)-4	Способен использовать современные информационные технологии и программные средства при моделировании технологических процессов.
ОПК(У)-5	Способен разрабатывать нормативно-техническую документацию, связанную с профессиональной деятельностью с учетом стандартов, норм и правил.
ОПК(У)-6	Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий.
ОПК(У)-7	Способен разрабатывать современные экологичные и безопасные методы рационального использования сырьевых и энергетических ресурсов в машиностроении.
ОПК(У)-8	Способен оптимизировать затраты на обеспечение деятельности производственных подразделений.
ОПК(У)-9	Способен разрабатывать и осваивать новое технологическое оборудование.
ОПК(У)-10	Способен разрабатывать методики контроля и обеспечения производственной и экологической безопасность на рабочих местах.
ОПК(У)-11	Способен организовывать разработку и применение алгоритмов и современных цифровых программных методов расчетов и проектирования отдельных устройств и подсистем мехатронных и робототехнических систем с использованием стандартных исполнительных и управляющих устройств, средств автоматики, измерительной и вычислительной техники в соответствии с техническим заданием, разрабатывать цифровые алгоритмы и программы управления робототехнических систем.
ОПК(У)-12	Способен организовывать монтаж, наладку, настройку и сдачу в эксплуатацию опытных образцов мехатронных и робототехнических систем, их

Код компетенции	Наименование компетенции
	подсистем и отдельных модулей.
ОПК(У)-13	Способен использовать основные положения, законы и методы естественных наук и математики при формировании моделей и методов исследования мехатронных и робототехнических систем.
ОПК(У)-14	Способен организовывать и осуществлять профессиональную подготовку по образовательным программам в области машиностроения.
Профессиональные компетенции	
ПК(У)-1	Способен обрабатывать и анализировать научно-техническую информацию, планировать и проводить эксперименты и оформлять результаты исследований и разработок.
ПК(У)-2	Готов к оформлению элементов документации, проектов планов и программ проведения отдельных этапов работ.
ПК(У)-3	Способен к разработке рабочей проектно-конструкторской и эксплуатационной документации РТС в соответствии с требованиями нормативной документации.
ПК(У)-4	Готов разработать программное обеспечение изделий РТС робототехники.

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа – Инженерная школа информационных технологий и робототехники
Направление подготовки – 15.04.06 Мехатроника и робототехника
Отделение школы (НОЦ) – Отделение автоматизации и робототехники

УТВЕРЖДАЮ:
Руководитель ООП

_____	_____	Филипас А.А.
(Подпись)	(Дата)	(Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

Обучающемуся:

Группа	ФИО
8ЕМ11	Иванов Егор Андреевич

Тема работы:

Разработка программно-аппаратного комплекса для 4-х тросовой системы обезвешивания	
Утверждена приказом директора (дата, номер)	№ 38-110/с от 07.02.2023

Срок сдачи студентом выполненной работы:	02.06.2023

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе *(наименование объекта исследования или проектирования; производительность или нагрузка; режим работы (непрерывный, периодический, циклический и т. д.); вид сырья или материал изделия; требования к продукту, изделию или процессу; особые требования к особенностям функционирования (эксплуатации) объекта или изделия в плане безопасности эксплуатации, влияния на окружающую среду, энергозатратам; экономический анализ и т. д.).*	1) Установка тросового обезвешивания; 2) Три режима работы – сервисный режим, режим независимого управления и режим пространственного управления объектом; 3) Наличие программного обеспечения пользователя;
Перечень подлежащих исследованию, проектированию и разработке вопросов *(аналитический обзор по литературным источникам с целью выяснения достижений мировой науки техники в рассматриваемой области; постановка задачи исследования, проектирования, конструирования; содержание процедуры исследования, проектирования, конструирования; обсуждение результатов выполненной работы; наименование дополнительных разделов, подлежащих разработке; заключение по работе).*	1) Обзор существующих решений; 2) Доработка аппаратной составляющей установки; 3) Разработка программного обеспечения микроконтроллера; 4) Разработка программного обеспечения пользователя; 5) Разработка вспомогательного программного обеспечения; 6) Разработка протоколов для взаимодействия; 7) Проведение исследования поведения готовой системы на предмет точности позиционирования.

Перечень графического материала *(с точным указанием обязательных чертежей)*	Структурная схема аппаратной части установки обезвешивания Общая архитектура программного комплекса Архитектура программы микроконтроллера APP1 Общая архитектура основного программного обеспечения Архитектура модуля управления

Консультанты по разделам выпускной квалификационной работы
(с указанием разделов)

Раздел	Консультант
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Доцент ОСГН ШБИП Былкова Татьяна Васильева
Социальная ответственность	Профессор ООД ШБИП Федорчук Юрий Митрофанович
Раздел на иностранном языке	Доцент ОИЯ ШБИП Диденко Анастасия Владимировна

Названия разделов, которые должны быть написаны на русском и иностранном языках:

Обзор существующих решений

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	07.02.2023

Задание выдал руководитель и консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Заведующий кафедрой – руководитель отделения на правах кафедры ОАР ИШИТР	Филипас А.А.	к.т.н., доцент		07.02.2023
Старший преподаватель ОАР ИШИТР	Беляев А.С.	–		07.02.2023

Задание принял к исполнению обучающийся:

Группа	ФИО	Подпись	Дата
8ЕМ11	Иванов Егор Андреевич		07.02.2023

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа – Инженерная школа информационных технологий и робототехники
Направление подготовки – 15.04.06 «Мехатроника и робототехника»
Уровень образования – Магистратура
Отделение школы (НОЦ) – Отделение автоматизации и робототехники
Период выполнения – Весенний семестр 2022/2023 учебного года

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Обучающийся:

Группа	ФИО
8ЕМ11	Иванов Егор Андреевич

Тема работы:

Разработка программно-аппаратного комплекса для 4-х тросовой системы обезвешивания

Срок сдачи студентом выполненной работы:	02.06.2023

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
29.05.2023	Основная часть ВКР	60
30.05.2023	Раздел «Социальная ответственность»	20
30.05.2023	Раздел «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»	20

СОСТАВИЛ:

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Заведующий кафедрой – руководитель отделения на правах кафедры ОАР ИШИТР	Филипас А.А.	к.т.н., доцент		07.02.2023

Консультант

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Старший преподаватель ОАР ИШИТР	Беляев А.С.	–		07.02.2023

СОГЛАСОВАНО:

Руководитель ООП

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Заведующий кафедрой – руководитель отделения на правах кафедры ОАР ИШИТР	Филипас А.А.	к.т.н., доцент		07.02.2023

Обучающийся

Группа	ФИО	Подпись	Дата
8ЕМ11	Иванов Егор Андреевич		07.02.2023

РЕФЕРАТ

Выпускная квалификационная работа 125 с., 50 рис., 44 табл., 22 источника, 1 прил.

Ключевые слова: система тросового обезвешивания, программное обеспечение, встраиваемые системы, теория автоматического управления.

Объектом исследования является установка тросового обезвешивания.

Цель работы – разработка программно-аппаратного комплекса системы четырех тросового обезвешивания и исследование его работы применимо к обезвешиванию объектов.

В процессе исследования был проведен обзор существующих решений систем тросового обезвешивания. Была выполнена доработка аппаратной составляющей системы тросового обезвешивания. Была выполнена разработка комплекса программного обеспечения, включающего в себя микроконтроллерные программы, программы для персонального компьютера, а также протоколы для межпрограммного взаимодействия. В итоге работы был проведен ряд экспериментов для оценки точности работы системы.

Финансовая эффективность проекта показала, что затраты на разработку данной системы на 11% ниже, чем у аналогов.

В будущем планируется доработка полученной системы и дальнейшее проведение исследований, направленных не только на точность, но и на скорость отработки.

Содержание

Введение

Сегодня технологии развиваются со все большей и большей скоростью. Современные автоматизированные системы внедряются повсеместно и порой становятся не просто средством достижения комфорта, а необходимостью. Системы обезвешивания не стали исключением.

Сегодня системы обезвешивания постепенно внедряются в различные сферы. Среди областей применения можно выделить медицинскую отрасль для реабилитации пациентов с опорно-двигательными функциями, аэрокосмическую отрасль, где данные системы позволяют проводить эксперименты с имитацией невесомости, и спортивную сферу, и сферу развлечений.

Разработка подобных систем тросового обезвешивания представляет собой достаточно комплексную задачу, в которой необходимы навыки из различных сфер, таких как, теория управления, электроника, информатика и программирование, механика, что приводит к необходимости командной работы специалистов из различных сфер. В данной работе будет представлено исполнение системы тросового обезвешивания, включающей в себя четыре незваисимых троса, а также разработка программного обеспечения для данной системы.

Целью работы является разработка программно-аппаратного комплекса системы четырех тросового обезвешивания и исследование его работы.

Объектом исследования в работе является установка тросового обезвешивания.

Предметом исследования является разработка программного обеспечения и алгоритмов управления системой тросового обезвешивания.

Подобные системы тросового обезвешивания применимы и используются в различных отраслях, что доказывает наличие **практической значимости у данной выпускной квалификационной работы**.

1 Основная часть

1.1 Обзор существующих решений

Сегодня системы тросового обезвешивания активно используются в различных областях. Довольно широкое распространение данные системы получили и в медицинской отрасли. Там данные системы используются для реабилитации пациентов с травмами опорно-двигательного аппарата. Подход при разработке подобных систем схожий, однако наблюдаются некоторые различия. Ниже представлены некоторые из систем тросового обезвешивания.

Одной из подобных систем является ZeroG от компании Aretech [1] (рисунок 1.1). Данная реабилитационная система компенсации веса основана на однозвенном манипуляторе. При помощи одного троса, прикрепленного к заранее установленной рельсе, данная система позволяет пациенту перемещаться в заданной области.

Рисунок 1.1 – Система ZeroG (Aretech)

Другой подобной системой является Vector от компании Bioness [2] (рисунок 1.2). По своей сути и по устройству очень схожа с системой ZeroG. У данной системы также имеется рельса, вдоль которой происходит перемещение каретки с прикрепленным к ней тросом.

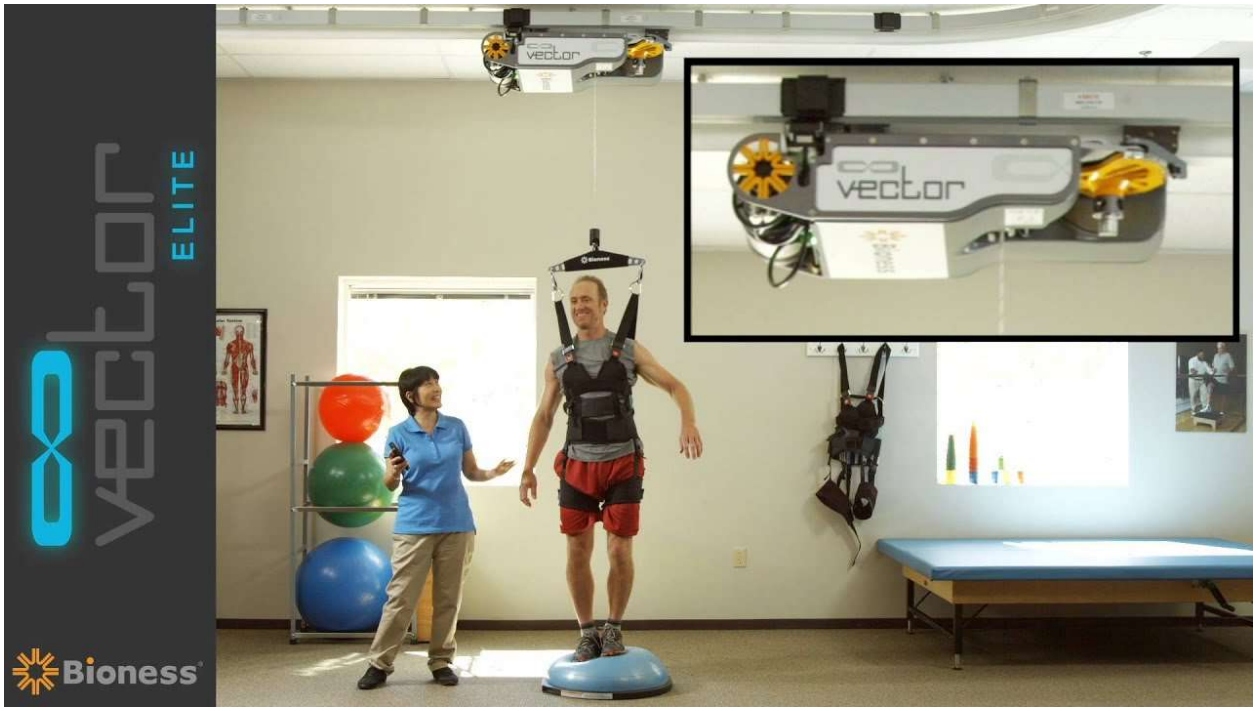


Рисунок 1.2 – Система Vector (Bioness)

Еще одной системой тросового обезвешивания является система Float [3] (рисунок 1.3). Данная система, в отличии от выше упомянутых, включает в себя четыре гибких троса. Данные тросы подсоединены к подвижным роликам, перемещающимся по закрепленным на потолке параллельным рельсам. Передвижение роликов происходит за счет приводов.

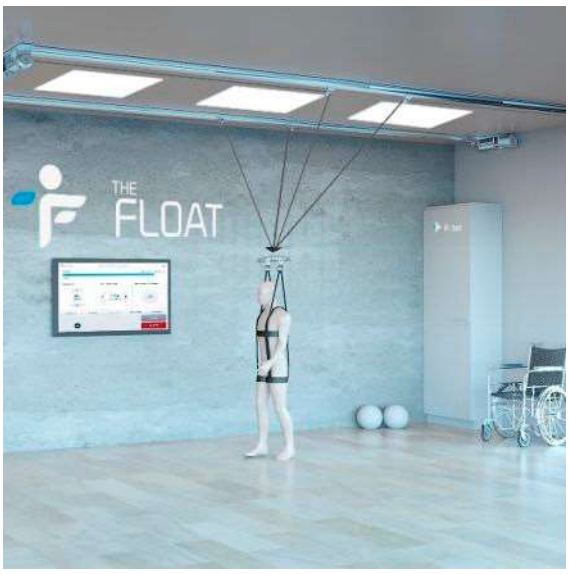


Рисунок 1.3 – Система Float

Также существует система Ryzen компании Motekmedikal [4] (рисунок 1.4). Данная система представляет собой аналогичную Float систему. Также имеется 4 троса, закрепленных через ролики, которые передвигаются по параллельным рельсам при помощи приводов.

Рисунок 1.4 – Система Ryzen (Motekmedikal)

Все вышеупомянутые системы представляют собой самые популярные на текущий момент системы тросового обезвешивания, применяемые в медицине.

Недостатками первых двух систем являются значительные ограничения в передвижении, так как для корректного обезвешивания объект должен передвигаться строго под рельсой.

У последних двух систем можно выделить небольшой недостаток, которым является фиксированная длина троса. Таким образом, активная область ограничивается непосредственно высотой расоположения объекта – чем объект выше – тем больше будет расстояние между двумя тросами на одной рельсе и тем меньше будет активная рабочая зона всей системы.

1.2 Обзор установки тросового обезвешивания

1.2.1 Исходное состояние установки и дополнения аппаратной части

На момент начала работ коллективом ОАР ИШИТР уже была выполнена часть аппаратной базы. Общий вид всей установки на момент начала работ представлен на рисунке 1.5.

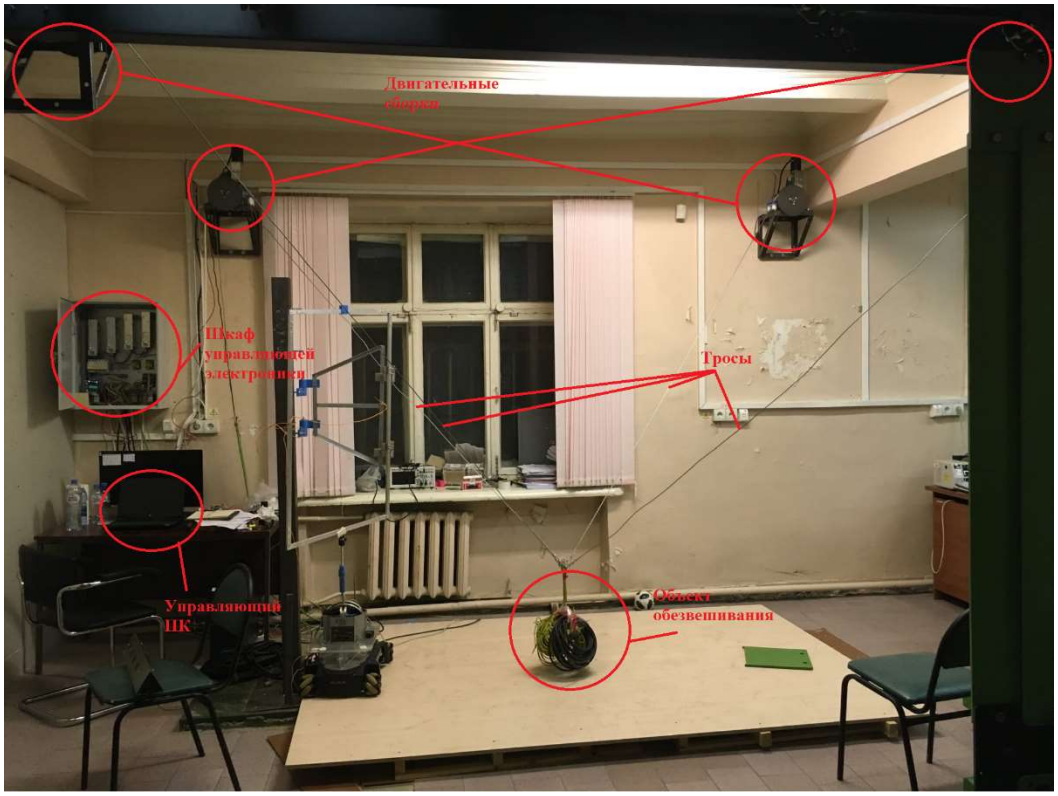


Рисунок 1.5 – Общий вид системы тросового обезвешивания на момент начала работ

Исходная система включала в себя следующее:

1) Двигатели постоянного тока. На валу каждого из двигателей был закреплен оптический квадратурный энкодер для отслеживания положения вала двигателя. Также на валу двигателя была закреплена бобина, на которую был намотан трос;
2) Шкаф управляющей электроники. Внутри шкафа были размещены блоки питания для получения питающих напряжений +24 и +5 Вольт, а также были размещены колодки и автоматика;
3) Персональный компьютер для разработки программного обеспечения;
4) Кабель-каналы и проложенная в них проводка для подключения всех электронных компонентов.

Выше описанная установка представляла собой скелет, на базе которого, для возможности работы с установкой, было необходимо внести дополнения к аппаратной части.

Самым важным дополнением стала установка микроконтроллера. Поскольку разработка отдельной платы для всей системы заняла бы довольно много временных ресурсов, было решено использовать отладочную плату на базе STM32 [5]. Для данной отладочной платы была выполнена интерфейсная плата с выводными колодками, которая обеспечивала удобство ее подключения к другим аппаратным модулям установки.

Также были разработаны и установлены платы энкодера и датчика тока. Принципиальная схема данной платы была представлена на рисунке 1.6. Она включает в себя подтяжку каналов энкодера к питанию, а также на ней располагается датчик тока для измерения тока через обмотки двигателя. Энкодер является квадратурным. Датчик тока представляет собой датчик на эффекте холла. Два вывода включаются в цепь питания двигателя, а с аналогового вывода происходит считывание сигнала 0-5 В, по которому через линейную Вольт-Амперную характеристику высчитывается значение тока. Датчик функционирует в пределах -50 – +50 А.

Также дополнением стал модуль энергонезависимой памяти EEPROM. Данный модуль необходим для сохранения актуального положения двигателей. Он взаимодействует с контроллером через интерфейс I2C и имеет встроенные 256 байт памяти, разбитых на 16 страниц по 16 байт каждая.

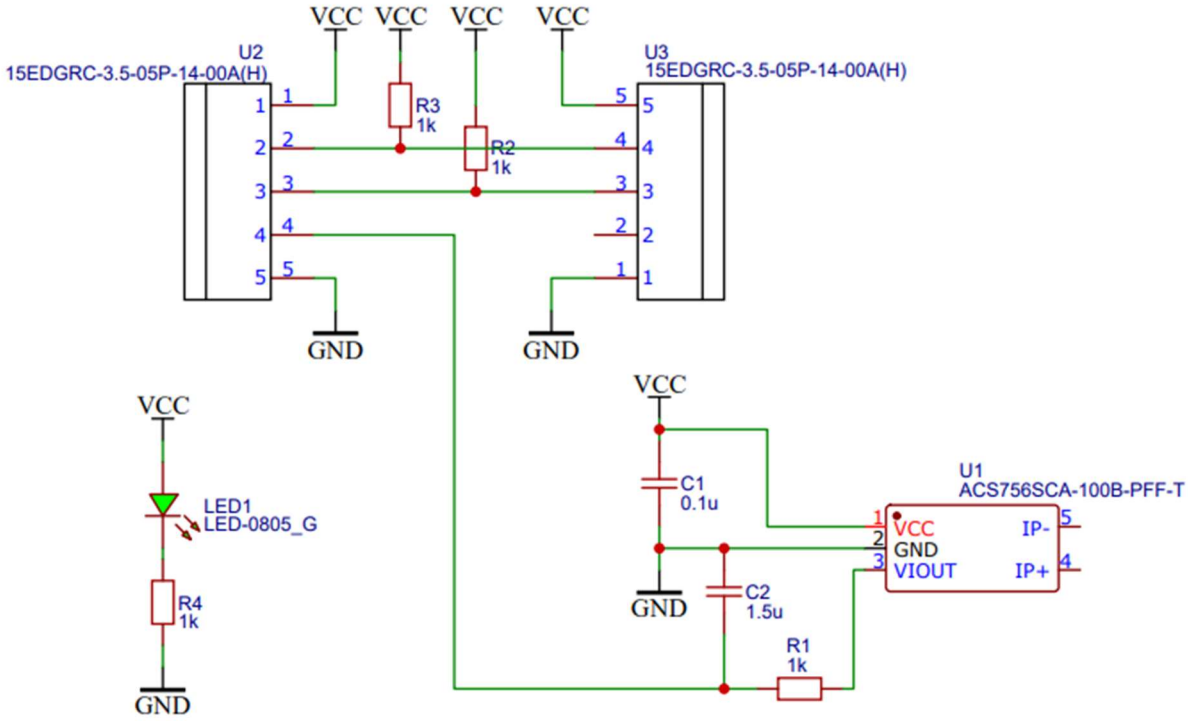


Рисунок 1.6 – Принципиальная схема платы датчиков

Последним дополнением стала установка модуля преобразователя UART-USB для возможности обмена данными с микроконтроллером со стороны персонального компьютера. Выбор в качестве интерфейса обмена данными UART был обусловлен отсутствием на отладочной плате распаянных USB или Ethernet интерфейсов.

В итоговом исполнении готовая к разработке программного обеспечения аппаратная установка имеет вид, представленный на структурной схеме на рисунке 1.7.

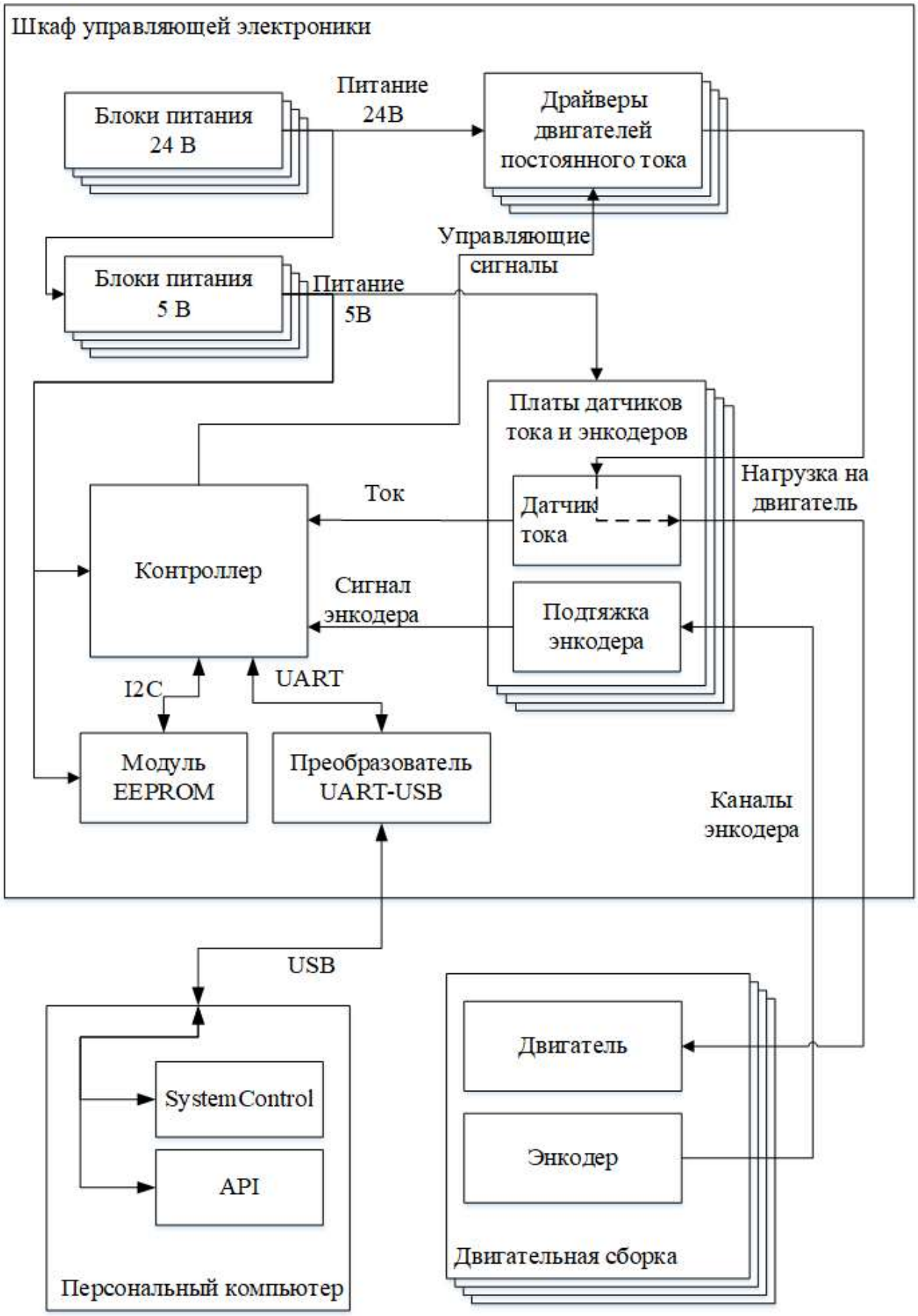


Рисунок 1.7 – Структурная схема конечного варианта исполнения аппаратной части установки обезвешивания

Имея данное аппаратное обеспечение становится возможным перейти к разработке программного обеспечения.

1.2.2 Программная часть установки

Программная составляющая установки делится на три модуля: программное обеспечение микроконтроллера, основное программное обеспечение ПК, а также вспомогательное программное обеспечение ПК.

Программное обеспечение микроконтроллера включает в себя две программы:

1) Программа загрузчика микроконтроллера (BOOT) – обеспечивает реализацию архитектуры хранения множества программ на едином микроконтроллере. Данная программа делит FLASH-память микроконтроллера на разделы и обеспечивает работу с ними извне: считывание прошивки в разделе, загрузка прошивки в раздел, очистка раздела, загрузка из раздела, подсчет контрольной суммы прошивки раздела. Данный подход необходим по следующим причинам. Во-первых, архитектурное разделение на отдельные программы позволяет полностью изолировать ответственность разных программ, что упрощает разработку программного обеспечения. Во-вторых, при необходимости обновления программного обеспечения уходит необходимость использования низкоуровневого интерфейса для прошивки необходимого раздела, что исключает возможность случайного затирания одного из разделов;
2) Программа управления двигателями (APP1) – обеспечивает управление двигателями постоянного тока, а также позволяет конфигурировать систему управления, считывать показания датчиков тока и энкодеров. В данном ПО возможно задавать уставки на двигатели, конфигурировать параметры регуляторов, считывать ток, скорость и положение, выполнять калибровку положения валов двигателей.

Основное программное обеспечение ПК включает в себя единственную программу, состоящую из трех модулей:

1) Модуль загрузчика (Bootloader) – модуль, взаимодействующий с программой BOOT микроконтроллера. Данный модуль представляет собой пользовательский интерфейс для сервисного обслуживания установки (обновления программного обеспечения, его верификации, переходов между программами микроконтроллера);
2) Модуль управления (Control) – модуль, взаимодействующий с программой APP1 микроконтроллера. Данный модуль представляет собой пользовательский интерфейс для управления как отдельными двигателями установки, так и пространственным перемещением объекта обезвешивания;
3) Модуль блочного программирования (Block) – модуль, который призван взаимодействовать с программой APP2 микроконтроллера, которая на текущий момент находится в разработке. Данный модуль создан исключительно в учебных целях и представляет собой среду блочного программирования, похожую на Simulink, для управления установкой без написания программного кода.

Вспомогательное программное обеспечение ПК является внешним комплексом программного обеспечения, которое не встроено в основное программное обеспечение. Оно включает в себя следующие программы:

1) Программное обеспечение для определения положения объекта обезвешивания (Trajectory Tracker) – определяет пространственные координаты объекта обезвешивания посредством камеры и закрепленного на объекте ArUco маркера. Разработано для оценки точности перемещения объекта в пространстве;
2) Программное обеспечение для задания траекторий передвижения обезвешиваемого объекта в пространстве (Trajectory Builder) – представляет собой математический интерпретатор с графической оболочкой для создания и экспорта целевой траектории в основное программное обеспечение. Необходим для разгрузки основного программного обеспечения от встраивания модуля создания сложных математических траекторий.

Общая архитектура взаимодействия программных комплексов и их модулей представлена на рисунке 1.8.

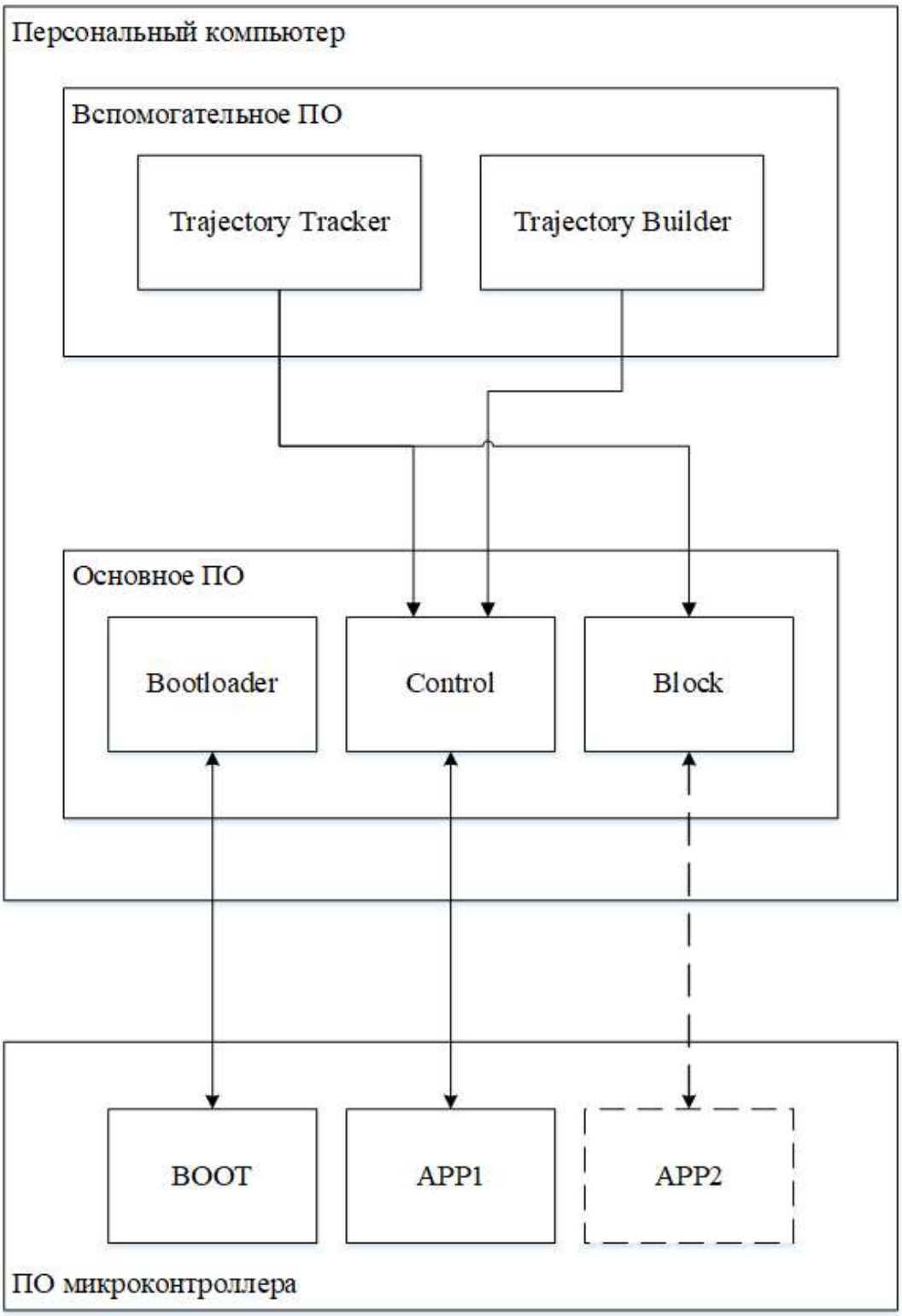


Рисунок 1.8 – Общая архитектура программного комплекса

Программа микроконтроллера APP2 отмечена пунктиром, так как данная программа находится в разработке и не внедрена в систему, а модуль Block основного программного обеспечения ПК является экспериментальным.

В общем случае, из архитектуры видно, что по своей сути каждый модуль основного программного обеспечения ПК является ответной частью для каждой из программ на микроконтроллере (BOOT – Bootloader, APP1 - Control).

Поскольку происходит взаимодействие между двумя устройствами – персональным компьютером и микроконтроллером, становится необходимостью формализация обмена данными между ними. Для формализации был разработан комплексный протокол взаимодействия, который включает в себя два протокола:

1) Basic Protocol – базовый протокол, который отвечает за сервисное обслуживание установки (обновление ПО, проверка ПО и т.д.) и отвечает за использование остальных протоколов;
2) Control Protocol – протокол управления двигателями.

1.3 Протоколы обмена данными с микроконтроллером

В системе имеется один сложный протокол, который имеет внешнюю и вложенную часть [6].

Внешней частью является протокол Basic Protocol. Данный протокол является базовым для всей системы. Каждая из программ микроконтроллера поддерживает его в большей или меньшей степени. Данный протокол выполняет сервисную задачу. С его помощью происходит запрос на перепрошивку какой-либо из программ микроконтроллера, проверка подключения установки к ПК и другие сервисные задачи. Помимо этого, данный протокол позволяет использовать внутри себя все остальные вложенные протоколы для взаимодействия с установкой.

Внутренняя часть представлена вложенным протоколом Control Protocol. Данный протокол индивидуален для программы APP1 микроконтроллера

и необходим для управления двигателями, калибровки, считывания показаний датчиков.

Подобная архитектура протоколирования была введена для возможности разделения задач различных программ микроконтроллера. С помощью введения данной архитектуры становится возможным гибкое внедрение новых протоколов, в качестве вложенных в Basic Protocol, или обновление старых, а также масштабирование системы протоколирования.

Далее будет представлено более подробное описание работы данных протоколов для лучшего понимания их работы.

1.3.1 Протокол Basic Protocol

Протокол Basic Protocol (далее – BP) является главным и самым внешним (по отношению к остальным) протоколом, он необходим для сервисного взаимодействия с микроконтроллером. Сервисные функции включают в себя чтение прошивки раздела, запись прошивки раздела, получения контрольной суммы прошивки раздела, переход между разделами, пинг контроллера, а также затирание прошивки в разделе. Кроме того, данный протокол включает в себя функцию передачи данных по вложенным протоколам. Все функции данного протокола (кроме поддержки перехода на внутренний протокол) полностью поддерживаются программой загрузчика микроконтроллера и частично поддерживаются другими программами.

Для большего понимания того, как работает данный протокол немного углубимся в то, каким образом происходит распределение встроенной в микроконтроллер FLASH-памяти.

Вся FLASH-память микроконтроллера полностью контролируется программой загрузчика. В ней расписано где начинается и заканчивается каждый из разделов. Схематичное распределение разделов в памяти представлено в таблице 1.1.

Таблица 1.1 – Схематичное расположение разделов во FLASH-памяти микроконтроллера

Номер раздела	0	1	2
Имя раздела	Загрузчик BOOT	Программа управления двигателями APP1	Программа для блочного программирования APP2 (раздел есть, но программа не реализована)
Начало раздела в памяти	0x8000000 (адрес)	APP_1_START_ADDRESS (макрос)	APP_2_START_ADDRESS (макрос)

Как видно, у каждого раздела есть свой номер, который хранится в программе загрузчика, а также свой адрес начала программы. Разделы хранятся в памяти последовательно друг за другом, поэтому вычислить то, какой размер памяти выделен для каждого раздела, можно путем вычисления разницы в адресах начала разделов.

В рамках протокола Basic Protocol у каждого раздела имеется свой ID. ID представляет собой 4-байтовую ASCII-строку. Фактически он представляет собой строку в 4 символа. У раздела загрузчика этот ID “BOOT”, у раздела программы управления двигателями “APP1”, а у еще не реализованной программы для блочного программирования он равен “APP2”. Этот ID отправляется в качестве ответа на команду пинга в протоколе Basic Protocol.

Теперь перейдем к описанию команд протокола и к описанию формата протокола.

Протокол Basic Protocol состоит из заголовка и полезных данных. Протокол является point-to-point протоколом и он синхронный. Это означает, что перед тем, как отправить новую команду, необходимо дождаться ответа на старую. Заголовок протокола представляет собой преамбулу для определения начала пакета, номера команды, а также размера данных, которые идут после заголовка. Структура заголовка протокола Basic Protocol представлена в таблице 1.2.

Таблица 1.2 – Структура заголовка протокола Basic Protocol

Преамбула	Номер команды	Размер данных после
2 байта Фиксированное значение (uint16_t) 0x5AA5	1 байт Номер команды из протокола (uint8_t)	4 байта Размер данных после заголовка в словах (слово = 32 бит) (uint32_t)

Поскольку обмен данными происходит в лабораторных условиях, в данный протокол не была заложена контрольная сумма, однако, для возможности контроля пакетов, была заложена преамбула. Формат размера данных после заголовка в словах обусловлен устройством микроконтроллера STM32, так проще производить прошивку микроконтроллера, так как прошивка происходит по словам, а не по байтам, кроме того, такой формат позволит избежать пользовательских ошибок при передаче количества байт прошивки, не кратных четырем.

В зависимости от команды данные после заголовка могут принимать различный формат. Далее опишем все команды протокола с примерами запросов / ответов.

В таблице 1.3 представлен полный набор команд протокола.

Таблица 1.3 – Команды протокола

Название	Номер HEX	Описание	Поддержка в разделах
PING	0x01	Пинг раздела	BOOT, APP1, APP2
VERIFY	0x02	Запрос контрольной суммы раздела	BOOT
JUMP	0x03	Переход в программу раздела	BOOT, APP1, APP2
READ	0x04	Считывание раздела	BOOT
WRITE	0x05	Запись прошивки в раздел	BOOT
ERASE	0x06	Очистка раздела	BOOT

CONTROL	0xF0	Переход во вложенный Control Protocol	APP1

Теперь пройдемся по каждой из команд отдельно

Команда PING

Команда для получения ID активной программы на микроконтроллере. Используется для идентификации того, в какой программе сейчас работает микроконтроллер, а также для проверки наличия устройства на линии. Формат запроса выглядит следующим образом.

2 байта	1 байт	4 байта
(uint16_t) 0x5AA5	(uint8_t) 0x01	(uint32_t) 0x00000000

Формат ответа микроконтроллера, работающего из программы в разделе BOOT имеет следующий вид.

2 байта	1 байт	4 байта	4 байта
(uint16_t) 0x5AA5	(uint8_t) 0x01	(uint32_t) 0x00000001	(uint8_t [4]) “BOOT”

Команда VERIFY

Команда для получения контрольной суммы раздела. Используется для сравнения прошивок. Возвращает 32-битную контрольную сумму раздела. Формат запроса для получения контрольной суммы первого раздела имеет следующий вид.

2 байта	1 байт	4 байта	4 байта
(uint16_t) 0x5AA5	(uint8_t) 0x02	(uint32_t) 0x00000001	(uint32_t) Номер раздела 0x00000001

Формат ответа микроконтроллера, контрольная сумма первого раздела которого равна 0x89ABCDEF имеет следующий формат.

2 байта	1 байт	4 байта	4 байта	4 байта
(uint16_t) 0x5AA5	(uint8_t) 0x02	(uint32_t) 0x00000002	(uint32_t) Номер раздела 0x00000001	(uint32_t) CRC32 0x89ABCDEF

Команда JUMP

Используется для перехода между программами различных разделов. Поскольку ни одна из программ не знает о том, в каком разделе она работает, то из всех программ, кроме загрузчика, возможно перейти только в нулевой раздел (раздел загрузчика, всегда нулевой). Из загрузчика возможно перейти в любой раздел. Формат запроса для перехода во второй раздел имеет следующий формат.

2 байта	1 байт	4 байта	4 байта
(uint16_t) 0x5AA5	(uint8_t) 0x03	(uint32_t) 0x00000001	(uint32_t) Номер раздела 0x00000001

Формат ответа от микроконтроллера, при успешном переходе имеет следующий вид.

2 байта	1 байт	4 байта	4 байта
(uint16_t) 0x5AA5	(uint8_t) 0x03	(uint32_t) 0x00000001	(uint32_t) Номер раздела 0x00000001

Команды READ и WRITE

В отличии от всех выше представленных команд, команды READ и WRITE устроены сложнее. Для понимания их работы необходимо пояснить способ, по которому происходит прошивка пакета данных внутри микроконтроллера. Для этого, опишем частный случай прошивки двоичного файла в микроконтроллер при помощи команды WRITE, а затем вычитывание раздела.

Как только мы закончили писать программный код и собрали прошивку из исходных файлов, у нас появляется бинарный файл. Данный файл имеет размер в несколько слов (слово = 4 байта). Поскольку всю прошивку микроконтроллера передавать за раз было бы не очень уместно из-за ограничений по оперативной памяти микроконтроллера, прошивка передается частями (чанками) фиксированного размера.

Внутри программы загрузчика микроконтроллера у каждого раздела имеется два типа счетчика. Первый – счетчик адреса прошивки раздела, второй – счетчик адреса считывания прошивки. Данные счетчики необходимы для возможности прошивки микроконтроллера без необходимости в командах начала и окончания прошивки. Каждый раз, когда на контроллер поступает команда прошивки раздела, контроллер забирает из пакета чанк в N слов, после чего начинает записывать этот чанк в раздел, начиная с того адреса, который сейчас хранится в счетчике адреса прошивки, инкрементируя его после записи каждого слова. Таким образом мы абстрагируемся от необходимости инициализации и окончания записи и чтения. В программе загрузчика также имеется таймер для фиксации таймаута записи / считывания прошивки. Данный таймер запускается каждый раз, когда происходит запрос на прошивку. Если в пределах истечения таймера происходит еще один запрос по прошивке, то таймер сбрасывается и запускается вновь. Если таймер истек, то сбрасывается счетчик адреса записи прошивки до значения адреса начала раздела. Таким образом, если у нас есть файл размером в 512 байт или 128 слов, и мы решили записать его в раздел номер 1 чанками по 16 слов, то необходимо разбить прошивку на 8 чанков по 16 слов и последовательно передавать ее. Как только прошивка закончится, то оставить передачу прошивки и после истечения таймера, контроллер поймет, что прошивка закончилась и сбросит счетчик. После этого, можно считать, что прошивка записана успешно. Полная аналогия происходит и с командой вычитывания прошивки, только используется счетчик адреса чтения.

В качестве примера рассмотрим запись в первый раздел и дальнейшее чтение выше упомянутого двоичного файла размером в 128 слов, размером по 16 слов за единый запрос.

Формат каждого запроса на запись чанка прошивки будет иметь следующий вид.

2 байта	1 байт	4 байта	4 байта	64 байта
(uint16_t) 0x5AA5	(uint8_t) 0x05	(uint32_t) 0x00000011 (16 слов + номер раздела)	(uint32_t) Номер раздела 0x00000001	(uint32_t [16]) Данные чанка …

После того, как запрос будет обработан на контроллере и будет записан чанк прошивки, придет ответ следующего формата.

2 байта	1 байт	4 байта	4 байта
(uint16_t) 0x5AA5	(uint8_t) 0x05	(uint32_t) 0x00000001	(uint32_t) Номер раздела 0x00000001

Данные запросы на запись необходимо будет отсылать до тех пор, пока не закончится файл прошивки. Иными словами, 8 раз в нашем примере.

После записи данную прошивку можно вычитать. Вычитаем ее таким же образом, чанками по 16 слов. Формат запроса будет следующим.

2 байта	1 байт	4 байта	4 байта	4 байта
(uint16_t) 0x5AA5	(uint8_t) 0x04	(uint32_t) 0x00000002	(uint32_t) Номер раздела 0x00000001	(uint32_t) Размер вычитываемого чанка 0x00000010

Каждый раз после такого запроса, при успешной обработке нам будет приходить ответ следуюшего формата.

2 байта	1 байт	4 байта	4 байта	64 байта
(uint16_t) 0x5AA5	(uint8_t) 0x04	(uint32_t) 0x00000011	(uint32_t) Номер раздела 0x00000001	(uint32_t [16]) Данные чанка …

Данные запросы на чтение так же необходимо будет повторить 8 раз для вычитывания прошивки размеров в 128 слов.

Команда ERASE

Команда используется для полной очистки раздела. Формат очистки раздела 1 будет иметь следующий формат.

2 байта	1 байт	4 байта	4 байта
(uint16_t) 0x5AA5	(uint8_t) 0x06	(uint32_t) 0x00000001	(uint32_t) Номер раздела 0x00000001

Формат ответа, при успешной очистке раздела имеет следующий вид.

2 байта	1 байт	4 байта	4 байта
(uint16_t) 0x5AA5	(uint8_t) 0x06	(uint32_t) 0x00000001	(uint32_t) Номер раздела 0x00000001

Команда CONTROL

Данная команда поддерживается только в программах разделов APP1, APP2. Она говорит о том, что после заголовка Basic Protocol будет идти заголовок Control Protocol, который необходимо обработать. Формат запроса для данной команды имеет следующий вид.

2 байта	1 байт	4 байта	Control Protocol
(uint16_t) 0x5AA5	(uint8_t) 0xF0	(uint32_t) Зависит от Control Protocol	(uint32_t […]) …

Формат ответа рассматривать смысла нет, более подробно с данной командой будет произведено ознакомление в разделе описания протокола Control Protocol далее в работе.

1.3.2 Протокол Control Protocol

Протокол Control Protocol (далее – CP) является протоколом, поддерживаемым только в программе раздела APP1. Данный протокол узко специализирован для управления двигателями, калибровкой положения двигателей, считывания показаний датчиков посредством взаимодействия с регистрами программы APP1, в том числе включает крайне узко специализированные функции для более компактного, быстрого и удобного забора данных посредством одного запроса.

У данного протокола имеется собственный заголовок. Во избежание лишней информации далее не будет отображаться заголовок внешнего протокола Basic Protocol, так как его содержание было описано выше. Общий формат для заголовка Control Protocol представлен в таблице 1.4.

Таблица 1.4 – Формат заголовка протокола Control Protocol

ID двигателя	Номер команды
4 байта (см. Таблицу 1.5)	4 байт (см. Таблицу 1.6)

Для каждого из четырех двигателей в системе имеется собственный ID. Кроме того, в рамках данного протокола возможно вычитывание единого параметра сразу для всех двигателей через глобальный ID. ID двигателей в протоколе Control Protocol представлены в таблице 1.5.

Таблица 1.5 – ID двигателей в протоколе

Макрос	Номер HEX	Описание
DRIVE_1	0x00	ID первого двигателя
DRIVE_2	0x01	ID второго двигателя
DRIVE_3	0x02	ID третьего двигателя
DRIVE_4	0x03	ID четвертого двигателя
GLOBAL	0x04	Глобальный ID

Команды протокола представлены в таблице 1.6.

Таблица 1.6 – Команды Control Protocol

Название	Номер HEX	Формат данных далее	Описание
WRITE_REG	0x01	*single_reg_data_t multi-ple_reg_data_t*	Команда записи регистра
READ_REG	0x02	*single_reg_data_t multi-ple_reg_data_t*	Команда чтения регистра
READ_PLOT	0x06	*wc_plottables_t*	Команда вычиты-вания регистров для построения графиков
CALIBRATE	0x0A	*float float[4]*	Команда задания текущей позиции
SET_ZERO	0x0B	-	Команда задания нуля
STOP_DRIVE	0x0C	-	Команда оста-новки двигателей

Все команды, в зависимости от ID, применимы как для работы с одним двигателем, так и для работы со всеми двигателями сразу.

Описание структур данных после заголовка представлены ниже.

Формат представления *single_reg_data_t* представлен в таблице 1.7. Таблица 1.7 представлена ниже.

Таблица 1.7 – Формат *single_reg_data_t*

Название	Тип	Описание
Регистр (reg)	uint32_t (4 байта)	Номер регистра
Данные (data)	float (4 байта)	Данные

Формат представления *multiple_reg_data_t* представлен в таблице 1.8.

Таблица 1.8 – Формат *multiple_reg_data_t*

Название	Тип	Описание
Регистр (reg)	uint32_t (4 байта)	Номер регистра
Данные (data[4])	float [] (16 байт)	Данные

Формат представления *wc_plottables_t* представлен в таблице 1.9.

Таблица 1.9 – Формат *wc_plottables_t*

Название	Тип	Описание
pos_sp	Float (4 байта)	Уставка по положению
pos_fb	Float (4 байта)	Текущее положение
spd_sp	Float (4 байта)	Уставка по скорости
spd_fb	Float (4 байта)	Текущая скорость
cur_sp	Float (4 байта)	Уставка по току
cur_fb	Float (4 байта)	Текущий ток на двигателе
output	Float (4 байта)	Выход с системы управления

Все общение с контроллером в рамках регистров строится на том, что внутри программы контроллера заложен набор регистров, которые можно считывать или записывать. У каждого из регистров есть свой номер и сконфигурированный параметр разрешения на чтение (RO), запись (WO) или сразу и на чтение, и на запись (RW). Общий набор регистров, используемых в программе контроллера представлен в таблице 1.10.

Таблица 1.10 – Общий набор регистров

Название регистра	Номер регистра HEX	Разрешение	Описание
TORQUE	0x00	RW	Момент возмущения
POS_SP	0x01	RW	Уставка по положению (рад)
POS_FB	0x02	RO	Текущее положение (рад)
POS_ACC	0x03	RO	Интегральный накопитель контура положения
POS_ACC_THRES	0x04	RW	Ограничитель интегрального накопителя контура положения
POS_PERR	0x05	RO	Предыдущая ошибка контура положения
POS_Kp	0x06	RW	Пропорциональный коэффициент
POS_Ki	0x07	RW	Интегральный коэффициент
POS_Kd	0x08	RW	Дифференциальный коэффициент
POS_IS_ACT	0x09	RW	Разрешение работы контура
SPD_SP	0x0A	RW	Уставка по скорости (рад/с)
SPD_FB	0x0B	RO	Текущая скорость (рад/с)

SPD_ACC	0x0C	RO	Интегральный накопитель контура скорости
SPD_ACC_THRES	0x0D	RW	Ограничитель интегрального накопителя контура скорости
SPD_PERR	0x0E	RO	Предыдущая ошибка контура скорости
SPD_Kp	0x0F	RW	Пропорциональный коэффициент
SPD_Ki	0x10	RW	Интегральный коэффициент
SPD_Kd	0x11	RW	Дифференциальный коэффициент
SPD_IS_ACT	0x12	RW	Разрешение работы контура
CUR_SP	0x13	RW	Уставка по току (А)
CUR_FB	0x14	RO	Текущий ток (А)
CUR_ACC	0x15	RO	Интегральный накопитель контура тока
CUR_ACC_THRES	0x16	RW	Ограничитель интегрального контура тока
CUR_PERR	0x17	RO	Предыдущая ошибка контура тока
CUR_Kp	0x18	RW	Пропорциональный коэффициент
CUR_Ki	0x19	RW	Интегральный коэффициент
CUR_Kd	0x1A	RW	Дифференциальный коэффициент
CUR_IS_ACT	0x1B	RW	Разрешение работы контура
OUTPUT	0x1C	RO	Выход с системы управления (В)
OUTPUT_THRES	0x1D	RW	Ограничение выхода с системы управления

Формат запросов-ответов в данном протоколе схож с форматом запросов-ответов в протоколе Basic Protocol. Рассмотрим примеры запросов-ответов для каждой из команд.

Команда WRITE_REG

Формат запроса на запись в регистр уставки по положению для первого двигателя и для всех двигателей представлены ниже, соответственно.

Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)	single_reg_data_t	
			reg 4 байт (uint32_t)	data 4 байт (float)
[…]	0x00 (DRIVE_1)	0x01 (WRITE_REG)	0x01 (POS_SP)	[…]
Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)	multiple_reg_data_t	
			reg 4 байт (uint32_t)	data[4] 16 байт (float [])
[…]	0x04 (GLOBAL)	0x01 (WRITE_REG)	0x01 (POS_SP)	[…]

Формат ответа на данные запросы.

Basic Proto-col	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)	single_reg_data_t	
			reg 4 байт (uint32_t)	data 4 байт (float)
[…]	0x00 (DRIVE_1)	0x01 (WRITE_REG)	0x01 (POS_SP)	[…]
Basic Proto-col	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)	multiple_reg_data_t	
			reg 4 байт (uint32_t)	data[4] 16 байт (float [])
[…]	0x04 (GLOBAL)	0x01 (WRITE_REG)	0x01 (POS_SP)	[…]

Команда READ_REG

Формат запроса на чтение уставки по скорости для второго двигателя и для всех двигателей представлены ниже, соответственно. (* – данные в этой области не важны)

Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)	single_reg_data_t	
			reg 4 байт (uint32_t)	data 4 байт (float)
[…]	0x01 (DRIVE_2)	0x02 (READ_REG)	0x0A (SPD_SP)	[*]
Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)	multiple_reg_data_t	
			reg 4 байт (uint32_t)	data[4] 16 байт (float [])
[…]	0x04 (GLOBAL)	0x02 (READ_REG)	0x0A (SPD_SP)	[*]

Формат ответа на данные запросы.

Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)	single_reg_data_t	
			reg 4 байт (uint32_t)	data 4 байт (float)
[…]	0x01 (DRIVE_2)	0x02 (READ_REG)	0x0A (SPD_SP)	[…]
Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)	multiple_reg_data_t	
			reg 4 байт (uint32_t)	data[4] 16 байт (float [])
[…]	0x04 (GLOBAL)	0x02 (READ_REG)	0x0A (SPD_SP)	[*]

Команда READ_PLOT

Формат запроса для чтения данных анализа системы регулирования из третьего двигателя и изо всех двигателей представлен ниже.

Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)
[…]	0x02 (DRIVE_3)	0x06 (READ_PLOT)

Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)
[…]	0x04 (GLOBAL)	0x06 (READ_PLOT)

Формат ответа имеет следующий вид.

Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)	wc_plottables_t 28 байт (float [7])
[…]	0x02 (DRIVE_3)	0x06 (READ_PLOT)	[…]
Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)	wc_plottables_t 112 байт (float [4][7])
[…]	0x04 (GLOBAL)	0x06 (READ_PLOT)	[…]

Команда CALIBRATE

Данная команда выполняет задание калиброванных значений по положению вала двигателя. Пользователь передает значения, которые хочет, чтобы двигатель выставил как текущее положение вала двигателя.

Формат запроса на калибровку по положению двигателя 4 и на калибровку сразу всех двигателей представлен ниже.

Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)	Значения новой уставки 4 байта (float)
[…]	0x03 (DRIVE_4)	0x0A (CALIBRATE)	[…]
Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)	Значения новых уставок 16 байт (float[4])
[…]	0x04 (GLOBAL)	0x0A (CALIBRATE)	[…]

Формат ответа при успешной калибровке идентичен запросу.

Команда SET_ZERO

Данная команда выполняет задание нуля калибровки. После получения данной команды счетчик положения двигателя сбрасывается до нуля.

Формат запроса для калибровки двигателя 4 и для калибровки всех двигателей выглядит следующим образом.

Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)
[…]	0x03 (DRIVE_4)	0x0B (SET_ZERO)
Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)
[…]	0x04 (GLOBAL)	0x0B (SET_ZERO)

Формат ответа при успешной калибровке абсолютно идентичен формату запроса.

Команда STOP_DRIVE.

Данная команда выполняет функцию остановки двигателя. После получения данной команды устройство задает в уставку по контуру по положению величину, равную текущему значению счетчика положения двигателя.

Формат запроса на остановку двигателя 1 и всех двигателей представлены ниже.

Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)
[…]	0x00 (DRIVE_1)	0x0C (STOP_DRIVE)
Basic Protocol	ID 4 байт (uint32_t)	CMD 4 байт (uint32_t)
[…]	0x04 (GLOBAL)	0x0C (STOP_DRIVE)

Формат ответа при успешном выполнении остановки абсолютно идентичен формату запроса.

На этом рассмотрение протоколов для взаимодействия с установкой завершено.

1.4 Программное обеспечение микроконтроллера

Как упоминалось ранее, программное обеспечение микроконтроллера включает в себя 2 программы: программу загрузчика (BOOT) и программу управления двигателями (APP1). Каждая из программ находится в своем разделе памяти. Данный подход позволяет выполнять независимую разработку программного обеспечения и сохранять независимость программ.

Рассмотрим подробнее каждую из программ микроконтроллера.

1.4.1 Программа загрузчика микроконтроллера (BOOT)

Программа загрузчика микроконтроллера является основной программой, которая располагается в разделе BOOT, поддерживает протокол Basic Protocol и осуществляет сервисные функции для управления памятью микроконтроллера.

Архитектура микроконтроллера включает в себя работу с обменом данными по UART, а также запись-чтение-очистку FLASH-памяти микроконтроллера.

Рассмотрим более подробно программу загрузчика микроконтроллера.

Внутри программы отдельно реализованы следующие функции (блок-схемы представлены на рисунках 1.9 и 1.10):

1) C_Ping – команда, возвращающая 4-символьную строку ID программы;
2) C_Jump – команда загрузки программы из другого раздела;
3) C_Verify – команда, возвращающая значение контрольной суммы раздела;
4) C_Erase – команда очистки раздела;
5) C_Write – команда записи чанка прошивки в заданный раздел;
6) C_Read – команда считывания чанка прошивки из заданного раздела.

Для каждого раздела внутри программы имеется структура *boot_partition_t*. Данная структура включает в себя счетчик записи / чтения прошивки, время последней записи / чтения, адрес начала раздела, номер сектора, где

начинается раздел, а также количество секторов, которые занимает раздел. Описание данной структуры представлено в таблице 1.11.

Таблица 1.11 – Описание структуры *boot_partition_t*

Название поля	Тип данных	Описание
address	uint32_t (4 байта)	Адрес начала раздела
sector	uint32_t (4 байта)	Сектор начала расположения
sector_size	uint32_t (4 байта)	Количество занимаемых секторов
write_address	uint32_t (4 байта)	Счетчик адреса записи
read_address	uint32_t (4 байта)	Счетчик адреса чтения
write_last_call	uint32_t (4 байта)	Время с момента последней записи в память
read_last_call	uint32_t (4 байта)	Время с момента последнего чтения из памяти

Далее представлены блок-схемы сервисных команд программы.

При выполнении команд C_Read и C_Write счетчики записи и чтения инкрементируются после записи или чтения каждого слова. Величина инкремента составляет 1 слово или 4 байта, то есть каждый раз значение счетчика увеличивается на 4.

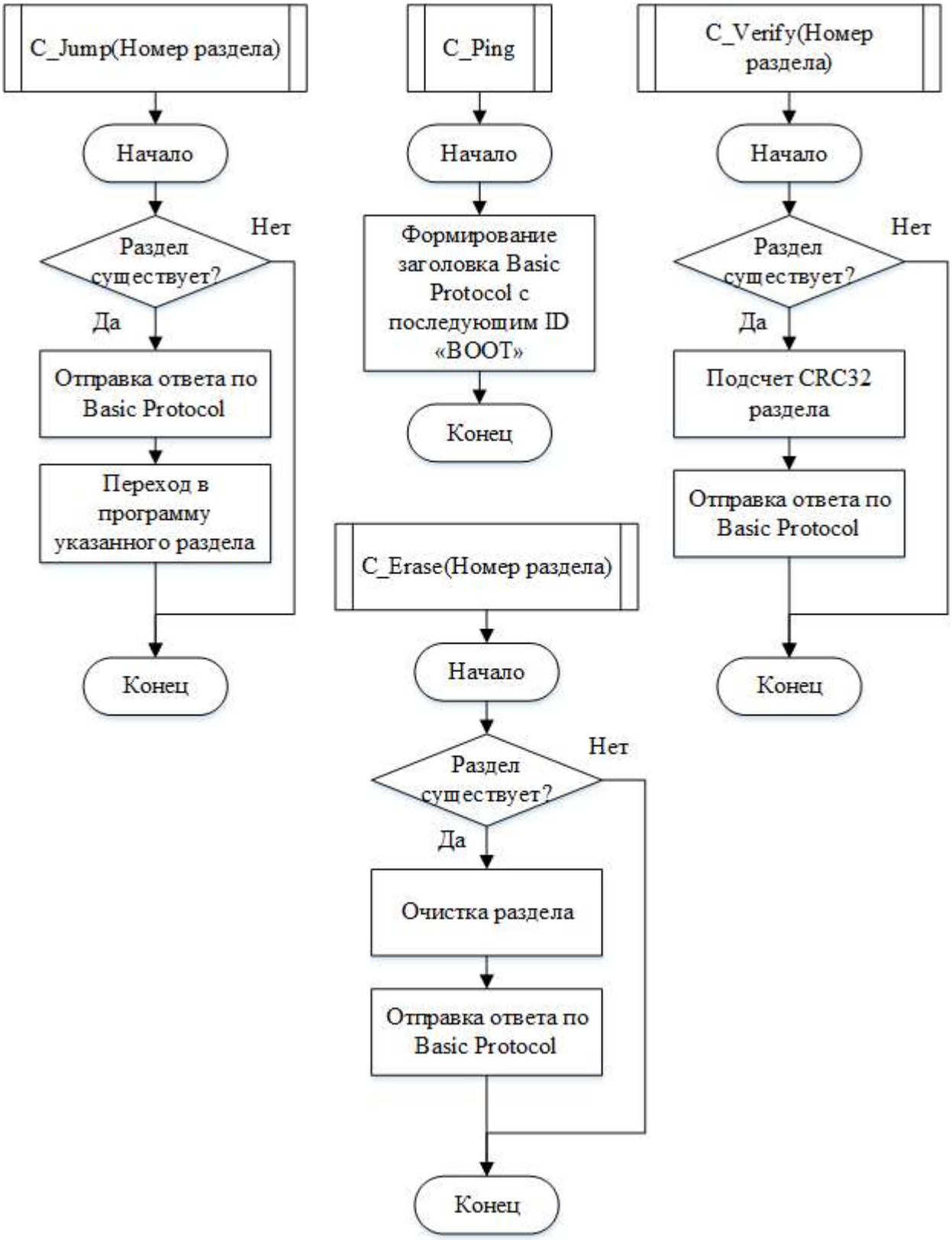


Рисунок 1.9 – Блок-схемы сервисных команд

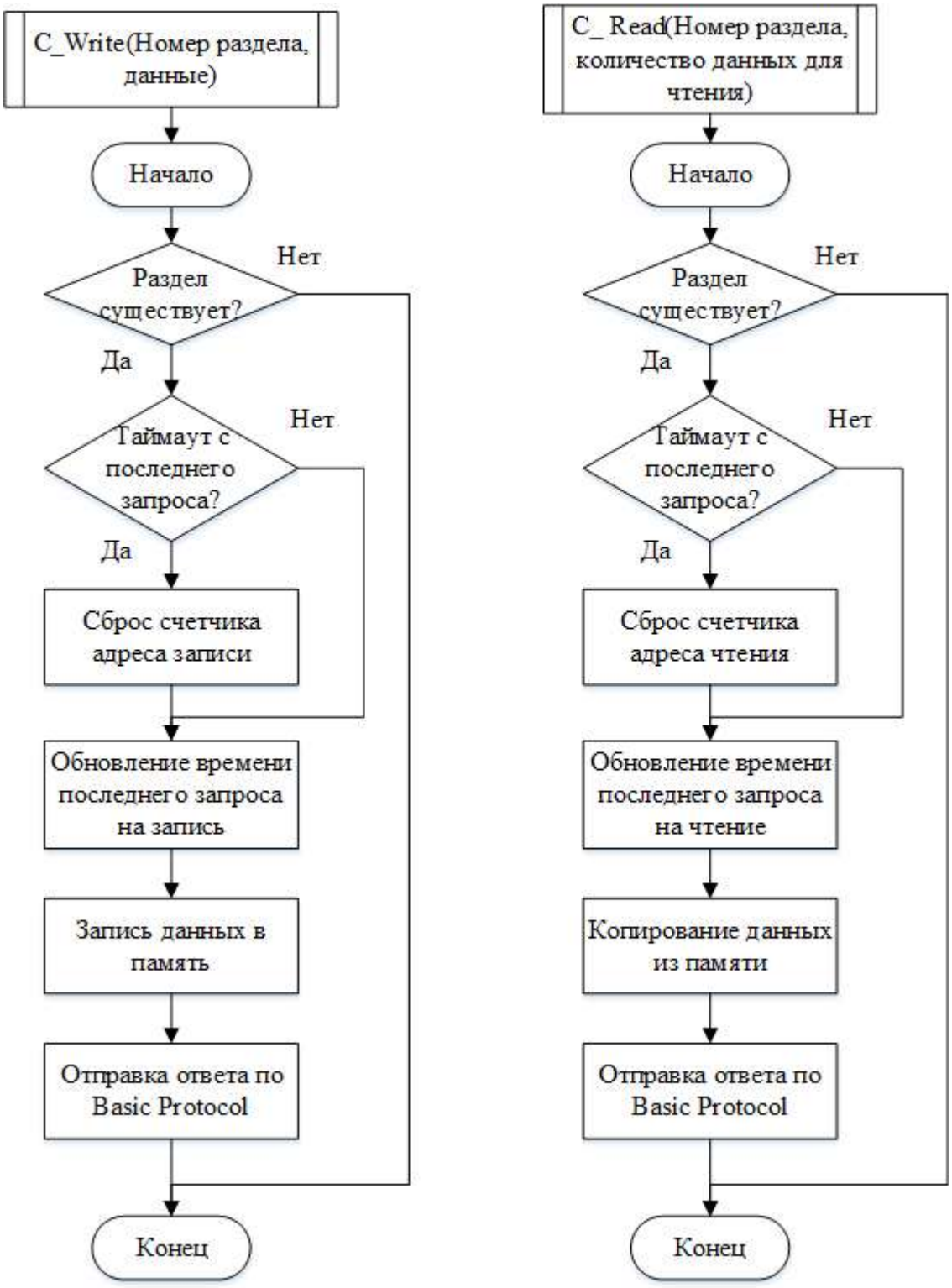


Рисунок 1.10 – Блок-схемы команд Read и Write

На основании этих команд строится вся логика работы программы. Основная программа состоит из одного бесконечного цикла, в котором происходит непрерывный вызов основной функции обработки протокола Basic Proto-

col, который реализован в функции Bootloader_Process(). Внутри данной функции, в зависимости от передаваемых на контроллер данных, происходит вызов базовых команд (C_Read, C_Write, C_Ping и т. д.). Общая блок-схема работы данной функции представлена на рисунке 1.11. Рисунок 1.11 представлен ниже.

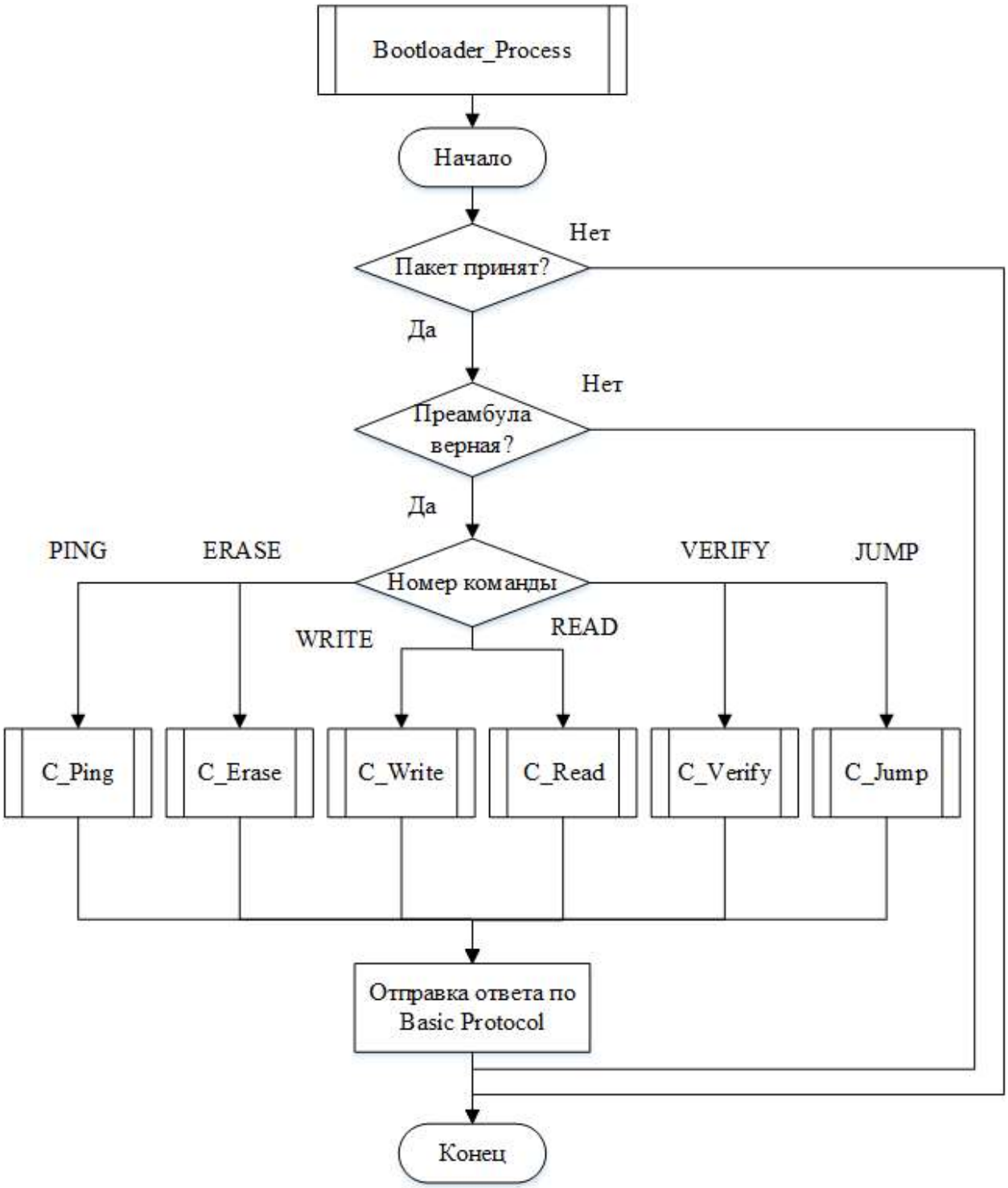


Рисунок 1.11 – Блок-схема работы команды Bootloader_Process

Вышеописанная блок-схема, заключенная в бесконечный цикл, полностью описывает работу программы загрузчика.

Стоит отметить, что данная программа является основной программой в микроконтроллере и переход в нее осуществляется непосредственно при подаче питания на микроконтроллер. Данная программа на текущий момент не может перепрошивать сама себя, прошивка программы загрузчика происходит непосредственно через программатор. В дальнейшем при усовершенствовании программы можно будет добавить копию программы загрузчика для возможности перезашивки загрузчика. Это может происходить следующим образом:

1) Поступает команда перепрошивки загрузчика, происходит копирование программы загрузчика в резервный раздел, после чего туда совершается переход, во FLASH памяти по определенному адресу выставляется флаг необходимости перепрошивки исходного раздела загрузчика;
2) Как только произошел переход отправляется новая программа загрузчика, которая начинает постепенно записываться в исходную область размещения программы загрузчика;
3) Как только прошивка завершается, происходит перезагрузка микроконтроллера и автоматически запускается новая версия программы загрузчика. Старая версия программы остается находиться там же, где она была до этого. Это позволяет, в случае неисправностей произвести откат к предыдущей версии.

Исходный код программы загрузчика для микроконтроллера STM32F429 можно увидеть по ссылке https://github.com/eai13/weight_control_system/tree/main_1/F429Port/f429boot или просканировав QR-код ниже.

1.4.2 Программа управления системой по ПИД-закону

Программа для управления системой по ПИД-закону является программой, хранящейся в разделе APP1 памяти микроконтроллера. Данная программа призвана обеспечить управление всей системой со всеми алгоритмами обработки, выполняющимися на низком уровне. В данную программу входят модули сохранения показаний энкодеров в энергонезависимой памяти, модули обработки энкодеров и датчиков тока, модуль обработки контуров систем управления, а также модуль частичной поддержки протокола Basic Protocol и полной поддержки Control Protocol.

Общая архитектура программного обеспечения представлена на рисунке 1.12. Рисунок 1.12 представлен ниже.

Данная архитектура представляет собой звезду. Узлом звезды является модуль регистров. В данный модуль происходит запись новых параметров, а также чтение из него. Все модули, включенные в данную систему, взаимодействуют исключительно с модулем регистров. Это позволяет значительно упростить систему и ввести абстракцию. Для понимания работы архитектуры пройдемся по каждому модулю по отдельности.

Используемая операционная система

Поскольку необходимо единовременно обрабатывать сразу несколько модулей, было принято решение ввести ОСРВ для абстракции от необходимости соблюдения таймингов вручную. Для этого была выбрана FreeRTOS [7]. Было введено 3 потока – по одному для модуля протоколирования, модуля энергонезависимой памяти, а также одна для обсчета ПИД-регуляторов [8].

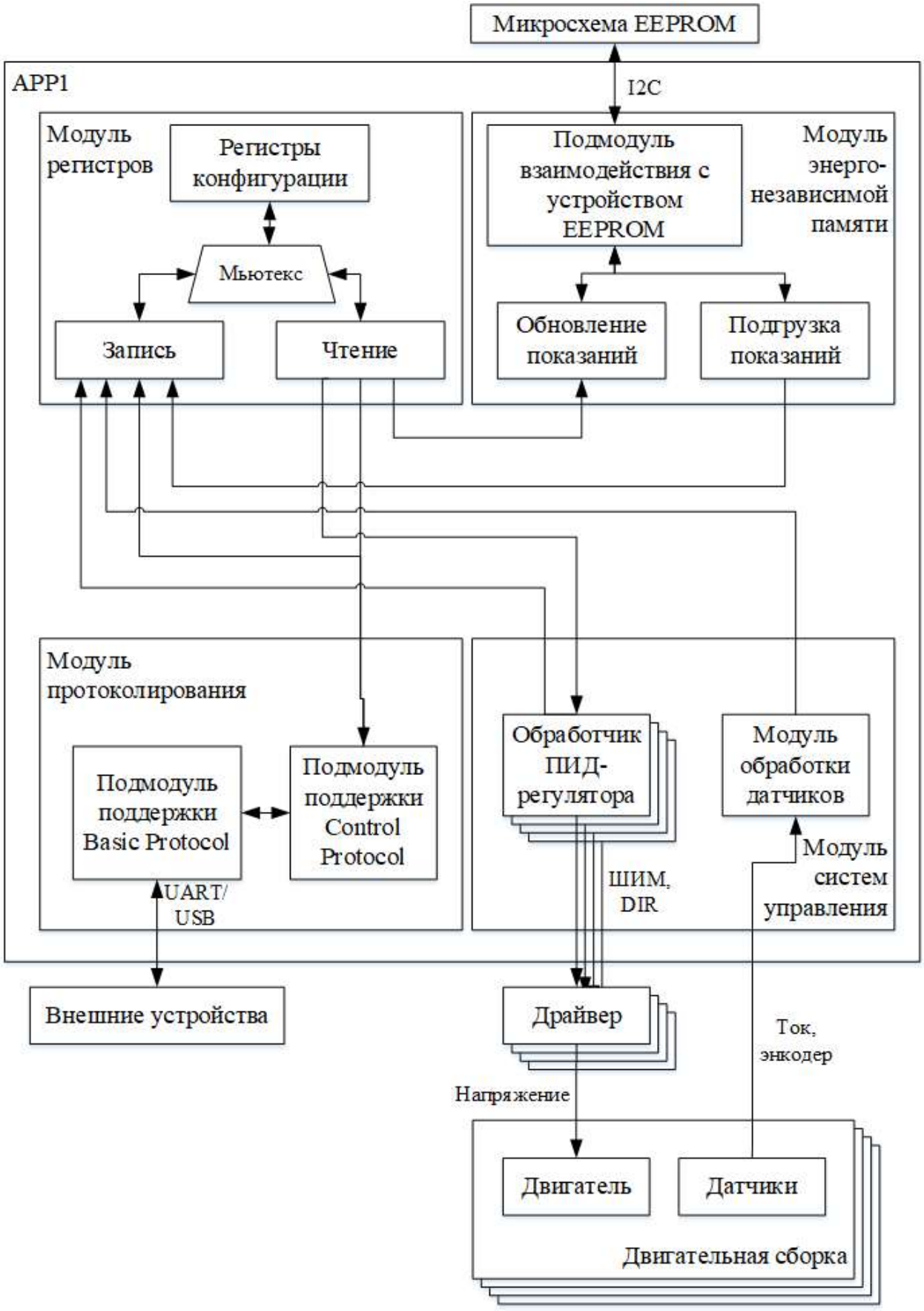


Рисунок 1.12 – Общая архитектура программы APP1

Модуль протоколирования

Данный модуль позволяет производить частичную поддержку Basic Protocol и полную поддержку Control Protocol. Из набора команд Basic Protocol поддерживаются следующие команды: PING, JUMP, CONTROL.

Подмодуль поддержки Basic Protocol по своей сути идентичен модулю из программы загрузчика. Отличаются они лишь набором команд. Подмодуль поддержки Control Protocol выполнен сходно с подмодулем Basic Protocol. У него также имеется готовый реализованный набор команд для записи и чтения регистров, которые вызываются в зависимости от команды по протоколу.

Поскольку система теперь состоит не из одного потока, а сразу из нескольких, необходимо ввести буферизацию для принимаемых и выходных данных. В противном случае есть риск перегрузить поток модуля протоколирования или упустить часть пакетов от клиента на линии. Для приема и передачи пакетов используется два кольцевых буфера и обработка UART в прерывании. В отличии от приема пакетов в блокирующем режиме в программе загрузчика, тут происходит непрерывный прием по одному байту и их складирование в кольцевой буфер приема. Далее, как только для потока модуля протоколирования выделяется время на работу, из кольцевого буфера приема вытаскивается некоторое количество байт и они обрабатываются. Передача производится примерно в таком же режиме. Как только произошла обработка входного пакета, формируется ответный пакет, который побайтово заносится в кольцевой буфер на отправку. Как только для потока модуля протоколирования выделяется свободное время, происходит отправка всего содержимого из кольцевого буфера на отправку.

Поскольку в таком режиме имеется возможность единовременного доступа к кольцевым буферам из потока и в прерывании, каждый из кольцевых буферов защищен мьютексом. Более подробно рассмотреть библиотеку кольцевого буфера можно увидеть по ссылке https://github.com/eai13/ecol/tree/main/ring_buffer или по QR-коду

Модуль регистров

Модуль регистров представляет собой выделенный форматированный участок оперативной памяти, где содержится информация о всех двигателях и их ПИД-регуляторах. Для каждого двигателя имеется следующая структура, в которой отражены все регистры. Данная структура представлена в таблице 1.12. Таблица 1.12 представлена ниже.

Таблица 1.12 – Структура информации о двигателях

Название поля	Тип данных	Описание
torque	*float_register_t* (5 байт)	Момент возмущения
position_l	*loop_t* (45 байт)	Контур положения
speed_l	*loop_t* (45 байт)	Контур скорости
current_l	*loop_t* (45 байт)	Контур тока
output	*float_register_t* (5 байт)	Выходное напряжение
output_thres	*float_register_t* (5 байт)	Ограничение выходного напряжения
encoder_s	*p_uint32_register_t* (5 байт)	Указатель на регистр счетчика таймера
current_s	*p_uint16_register_t* (5 байт)	Указатель на область, куда заносятся показания датчика тока
pwm_duty	*p_uint32_register_t* (5 байт)	Указатель на auto-reload register таймера, генерирующего ШИМ
dir_pins	*direction_control_t* (12 байт)	Структура с выводами для управления направлением вращения
direction	*wc_drive_dir_e* (1 байт)	Enum для быстрого получения последнего направления вращения

Для абстракции и удобства редактирования кода были заведены отдельные структуры регистров, в которые включены разрешения на запись и чтение регистров (*_*register_t*), на основании которых были заведены структуры для регистров, связанных с контурами регулирования (*loop_t*). Описание структуры *loop_t* представлено в таблице 1.13. Таблица 1.13 представлена ниже.

Таблица 1.13 – Структура регистров контуров управления

Название поля	Тип данных	Описание
sp	*float_register_t* (5 байт)	Величина уставки
fb	*float_register_t* (5 байт)	Величина обратной связи
acc	*float_register_t* (5 байт)	Интегратор для интегральной составляющей
acc_thres	*float_register_t* (5 байт)	Ограничение интегратора (anti-windup)
perr	*float_register_t* (5 байт)	Предыдущая ошибка регулирования для вычисления производной в Д-составляющей
Kp	*float_register_t* (5 байт)	Пропорциональный коэффициент
Ki	*float_register_t* (5 байт)	Интегральный коэффициент
Kd	*float_register_t* (5 байт)	Дифференциальный коэффициент
isActive	*float_register_t* (5 байт)	Флаг разрешения работы контура

Выше описанные регистры обновляются при очередной обработке ПИД-регуляторов для каждого двигателя.

Для абстракции работы с выводами выбора направления вращения была заведена структура, хранящая адрес порта и битовую маску вывода.

Для избежания единовременного доступа в память из двух потоков для каждого двигателя был выделен мьютекс. Чтение и запись происходят только при наличии свободного мьютекса, в противном случае происходит ожидание освобождения мьютекса.

Для записи и чтения используется две соответствующе названные команды: *PID_WriteReg* и *PID_ReadReg*.

Более подробно ознакомиться с устройством структуры регистров возможно по ссылке https://github.com/eai13/weight_control_system/blob/main_1/F429Port/WeightControl/Core/PID/pid.h или по QR-коду

Модуль энергонезависимой памяти

Поскольку разрабатываемая система может быть включена и выключена в любое время, а для ее естественного функционирования необходимо сохранение последнего перед выключением положения (от этого зависят выпущенные длины тросов), то было решено ввести внешний модуль энергонезависимой памяти EEPROM. На текущий момент в данном модуле хранятся последние положения двигателей перед сбросом питания с системы, однако в дальнейшем возможно ввести в нее и сохранение конфигурационных параметров.

Модуль EEPROM представляет собой 256 байт внешней энергонезависимой памяти, разделенной на 16 страниц по 16 байт каждая. Доступ к данному модулю осуществляется по интерфейсу I2C. Поскольку модуль является внешним и может быть заменен, для проверки корректности данных необходимо проверять его формат и, в случае некорректности форматирования, переформатировать его с занесением данных по умолчанию. Для этого используется четыре байта из первой страницы памяти. Там записывается 4-байтовая строка «MEMP», наличие которой говорит о том, что данный модуль отформатирован под наши требования и данные оттуда возможно использовать.

Также возможно внести туда контрольную сумму полезных данных, по которой говорить о том, корректны ли данные, однако на текущем этапе данная проработка внесена не была.

Начиная со второй страницы начинается область, где хранятся показания энкодеров. Каждое из значений энкодеров является показанием счетчика таймера и имеет формат uint32, что означает, что занимает 4 байта в памяти. Таким образом, для 4 значений показаний энкодеров используется вторая страница (16 байт).

Общее распределение памяти в модуле EEPROM показано в таблице 1.14. Таблица 1.14 представлена ниже.

Таблица 1.14 – Распределение памяти в модуле EEPROM

Адресация	Размер	Описание
0x00 – 0x03	4 байта	«MEMP»
0x03 – 0x0F	8 байт	Не используется
0x10 – 0x13	4 байта	Счетчик энкодера первого двигателя
0x14 – 0x17	4 байта	Счетчик энкодера второго двигателя
0x18 – 0x1B	4 байта	Счетчик энкодера третьего двигателя
0x1C – 0x1F	4 байта	Счетчик энкодера четвертого двигателя

Рассмотрим подробнее как происходит взаимодействие с данным модулем.

Как только программа APP1 запущена, происходит проверка наличия модуля, а также проверка правильности форматирования. Если форматирование верно, то происходит считывание счетчиков и занесение их показаний в структуру регистров. Если форматирования нет, то происходит форматирование памяти и туда заносятся значения по умолчанию.

Для работы данного модуля выделен отдельный поток. Данный поток работает на частоте в 0,25 герц и каждый раз при срабатывании обновляет показания счетчиков энкодеров.

Более подробно ознакомиться с данным модулем возможно по ссылке https://github.com/eai13/weight_control_system/tree/main_1/F429Port/WeightControl/Core/memory или по QR-коду

Модуль систем управления

Данный модуль является основой управления двигателями. Всего в системе имеется 4 двигателя. Каждый из них управляется по 3-контурной системе, которая включает в себя контур положения, скорости и тока. На выходе с каждой системы управления получается напряжение в Вольтах, которое переводится в значение скважности управляющего ШИМ-сигнала.

Данная система управления в качестве обратной связи использует показания датчиков тока и энкодеров.

Показания энкодеров считываются с помощью аппаратного модуля внутри таймера STM32. При реализации такого съема данных имеется небольшая тонкость. Поскольку значение, которое будет получаться является абсолютным, а счетчик таймера инициализируется нулем и имеет 16-битное разрешение, его необходимо привести в среднюю точку. В противном случае есть большой риск получить переполнение в обратную сторону и неадекватное поведение системы управления. Поскольку регистр счетчика 16-битный, то максимальное значение в нем будет иметь шестнадцатеричный вид 0xFFFF. Поэтому перевод в среднюю точку будет означать простую запись 0x7FFF в самом начале программы. Таким образом будет возможно использовать его для абсолютного счета положения при вращении в прямом и обратном направлении.

Показания датчиков тока снимаются с помощью встроенного в микроконтроллер АЦП. Поскольку качественный съем данных посредством АЦП процесс достаточно длительный и тратить на такую рутинную операцию время процессора нет необходимости, то для этого используется контроллер DMA. Это позволяет обеспечить автоматический перенос показаний АЦП со всех датчиков в необходимую область памяти. Для того, чтобы это происходило совсем без участия CPU был также задействован таймер, который обеспечивает периодический запуск АЦП и сбор данных через DMA. Сам по себе сигнал АЦП интерпретируется в пределах 0.00 – 5.12 Вольт. Датчик работает в пределах от -50 до +50 Ампер. Для вычисления показаний датчика используется ремаппинг напряжения в ток.

Перейдем к описанию работы контуров регулирования. Как уже упоминалось, каждый двигатель имеет в качестве системы управления 3-х контурную систему управления. Каждый из контуров имеет свой ПИД-регулятор. Для возможности более гибкой настройки имеется возможность отключения какого-либо из контуров скорости или тока, а также возможно изменять коэффициенты ПИД-регуляторов и ограничение интегратора.

Общая структура контуров регулирования представлена на рисунке 1.13. Рисунок 1.13 представлен ниже.

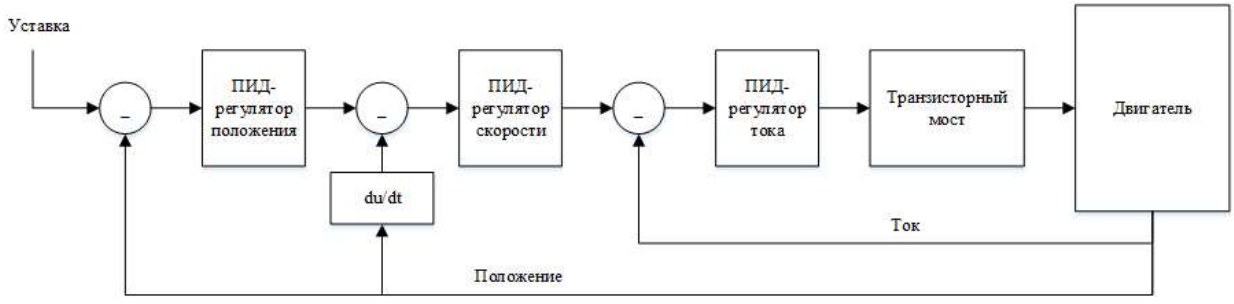


Рисунок 1.13 – Структура системы управления одним двигателем

Структура каждого ПИД-регулятора представлена на рисунке 1.14. Рисунок 1.14 представлен ниже.

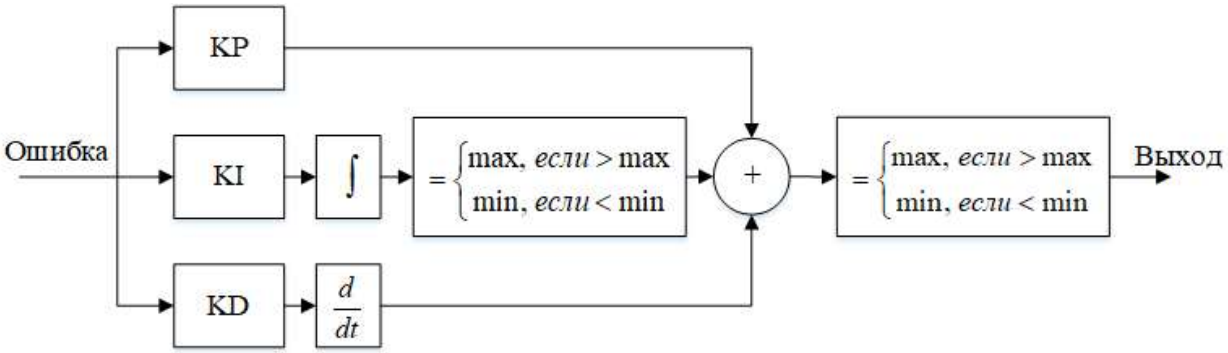


Рисунок 1.14 – Структура ПИД-регулятора

Для исключения насыщения интегральной составляющей в ПИД-регулятор введено ограничение интегральной составляющей (anti-windup). Обсчет системы управления каждого из двигателей производится в потоке, выделенном под модуль систем управления и имеет частоту 100 Гц.

Цифровая реализация такой системы управления описывается следующими формулами.

Сперва происходит запоминание старых параметров обратной связи для возможности взятия производной для Д-составляющей и получаем новые значения положения, скорости и тока

$$speed[0] = speed[1],\ position[0] = position[1],\ current[0] = current[1]$$

$$position[1] = new\_position$$

$$speed[1] = \frac{position[1] - position[0]}{T}$$

$$current[1] = new\_current$$

Далее сохраняем предыдущую ошибку регулирования, высчитываем текущую ошибку контура положения

$$error\_p[0] = error\_p[1]$$

$$error\_p[1] = position\_sp - position[1]$$

Высчитываем составляющие ПИД-регулятора контура положения

$$p\_position = error\_p[1] \cdot KP$$

$$i\_position = i\_position + error[1] \cdot KI$$

$$d\_position = \frac{error\_p[1] - error\_p[0]}{T}$$

Задаем ограничение на интегральную часть и высчитываем выход с ПИД-регулятора

$$if\ \ i\_position > i\_p\_Max$$
$$i\_position = i\_p\_Max$$
$$else\ \ if\ \ \ i\_position < i\_p\_Min$$
$$i\_position = i\_p\_Min$$
$$output\_position = p\_position + i\_position + d\_position$$
$$if\ \ \ output\_position > output\_p\_Max$$
$$output\_position = output\_p\_Max$$
$$else\ \ if\ \ \ output\_position < output\_p\_Min$$
$$output\_position = output\_p\_Min$$

Таким образом происходит обсчет ПИД-регулятора для контура положения. Аналогичным образом происходит обсчет контуров скорости и тока.

На выходе контура тока получается величина 0-24 Вольт, которая преобразуется в скважность ШИМ-сигнала

$$PWM = \frac{U}{U\_Max} \cdot PWM\_Max$$

где

PWM – скважность ШИМ,

U – полученное с системы управления напряжение,

U_Max – номинальное напряжение двигателя,

PWM_Max – период ШИМ.

Более подробно ознакомиться с данным модулем возможно по ссылке https://github.com/eai13/weight_control_system/tree/main_1/F429Port/WeightControl/Core/PID или по QR-коду

1.5 Программное обеспечение ПК

Программное обеспечение на ПК призвано предоставить пользователю готовую среду для организации работы всей установки. Программное обеспечение для ПК включает в себя основное приложение и набор вспомогательных. Связь установки и работа всего приложения завязана на отправке и приеме пакетов по протоколам Basic Protocol и Control Protocol. Далее будет подробно рассмотрен программный комплекс для ПК.

1.5.1 Основное программное обеспечение комплекса

Основное программное обеспечение включает в себя три базовых модуля. Данными модулями являются: модуль работы с загрузчиком (Bootloader), модуль управления системой через задание параметров встроенных регуляторов (Control), а также модуль, который реализован пока только на ПК, для возможности управления системой через компоновку блоков (Block).

Общая архитектура основного программного обеспечения представлена на рисунке 1.15. Рисунок 1.15 представлен ниже.

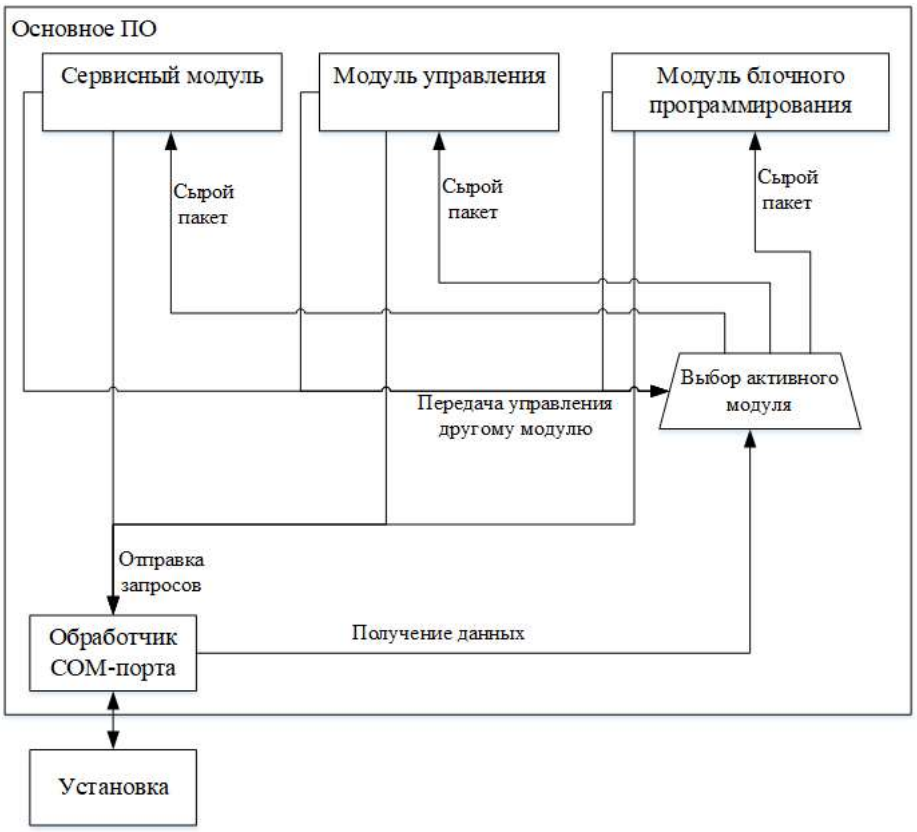


Рисунок 1.15 – Общая архитектура основного ПО

Разберем устройство каждого из модулей подробнее.

1.5.2 Сервисный модуль

Сервисный модуль предназначен для задач отладки. Через него возможно прошить в микроконтроллер только что собранную прошивку, считать какую-либо из программ, проверить ID активной на установке программы, очистить программу, а также перейти в нее.

Общий вид графического интерфейса сервисного модуля представлен на рисунке 1.16. Рисунок 1.16 представлен ниже.

Весь интерфейс сервисного модуля представлен шестью кнопками команд: «Ping», «Verify», «Erase», «Jump», «Write», «Read». Данные кнопки полностью повторяют набор команд из протокола Basic Protocol. Справа сверху расположено выпадающее окно выбора раздела, с которым хотим взаимодействовать при чтении, записи, проверке и т. д. прошивки. Сверху расположена строка для возможности выбора файла для прошивки файла в раздел или файл, куда считать прошивку из раздела.

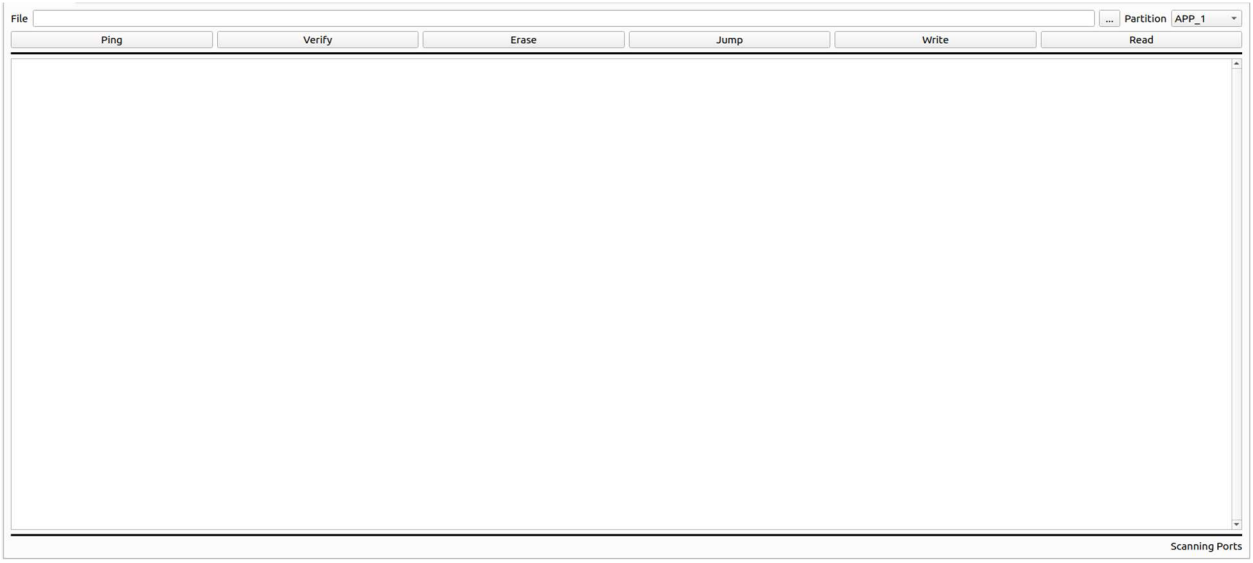


Рисунок 1.16 – Общий вид графического интерфейса сервисного модуля

Поскольку активных портов на компьютере может быть несколько, то в программу был добавлен автоматический поиск устройства. Для этого используется циклический перебор всех активных COM-портов и отправка на них

команды Ping. Если получен ответ на этот запрос и ID удовлетворяет, то выбирается данный порт и дальше происходит периодический фоновый запрос Ping для постоянной поддержки связи с установкой. В нижней части окна располагается строка состояния, в которой отображается активный порт для взаимодействия.

В центральной части окна располагается отладочная консоль. В ней отображается статус команд, которые отправляются нажатием соответствующей кнопки.

Данный модуль достаточно прост и архитектура у него проста. По нажатии кнопки происходит отправка пакета. При отправке пакета запускается таймер. Если у таймера сработал таймаут, то устройство пропало с линии и запускается циклический поиск устройства снова, пока оно вновь не появится на каком-либо из портов.

Интерфейс выполнялся таким образом, что в одном окне существуют сразу все модули. При этом в ПО заложено автоматическое переключение между модулями. Это решение было предпринято с целью автоматизации перехода между модулями при изменении активной программы на установке. Для этого используется фоновый Ping. В ответ на него каждый раз приходит ID активной программы установки. В зависимости от этого ID модуль автоматически может передать управление другому модулю и переключиться на него без участия пользователя. Таким образом пользователь сразу приступит к работе в другом модуле и не будет необходимости выяснять перешла ли установка в другую программу.

Таким образом, как только основное ПО запущено, происходит фоновый Ping и в ответ приходит ID «BOOT», что говорит о том, что установка находится в программе загрузчика и в ней необходимо оставаться. Как только пользователь выберет раздел APP1 и перейдет в него командой Jump, установка в ответ на фоновый Ping пришлет ID «APP1». Сервисный раздел поймет, что

установка перешла в программу раздела «APP1» и автоматически переключится на модуль управления. Такой механизм с фоновым Ping и автоматическим переключением реализован в каждом из модулей далее.

1.5.3 Модуль управления

Модуль управления необходим для управления установкой, находящейся в программе раздела «APP1». Данный модуль позволяет «покрутить» каждый из двигателей отдельно, просматривать в реальном времени графики с параметрами регулирования у каждого двигателя, а также строить 3D картину перемещения объекта обезвешивания и задавать траекторию перемещения, по которой должен проследовать объект.

Общий вид модуля управления представлен на рисунке 1.17. Рисунок 1.17 представлен ниже.

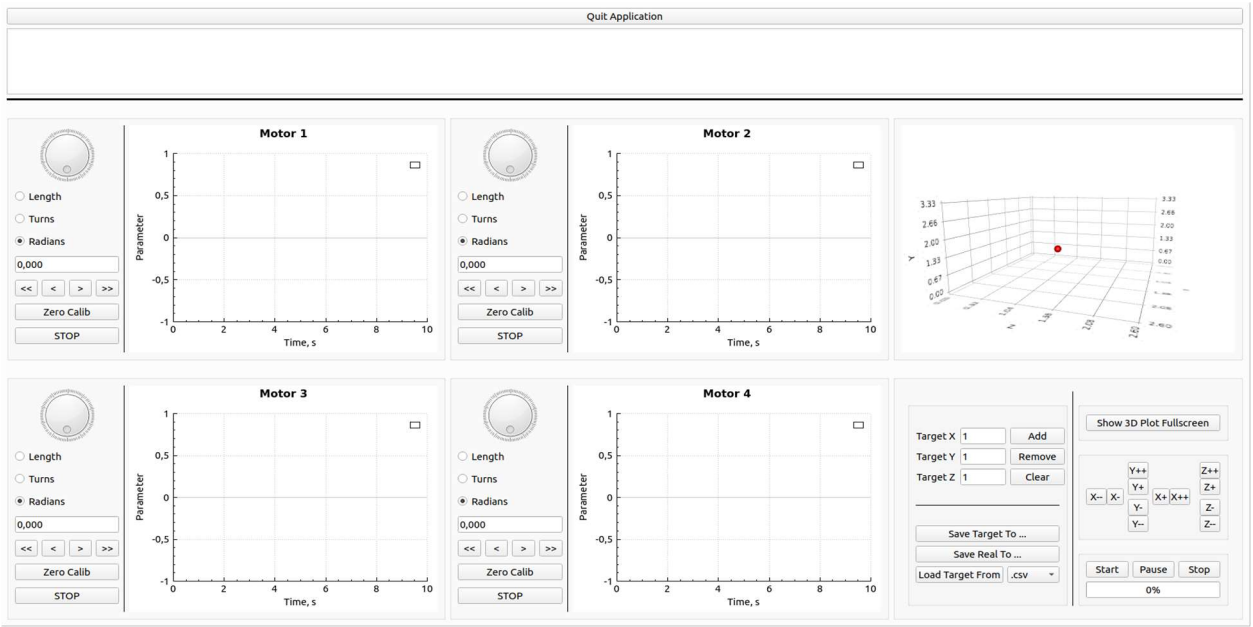


Рисунок 1.17 – Общий вид модуля управления

В самом верху располагается кнопка выхода из данного модуля и переход в сервисный модуль. Центральную и левую часть окна занимают подмодули независимого управления двигателями. Правая часть окна модуля занимается управлением перемещения объекта в пространстве.

Сперва рассмотрим модули независимого управления двигателями. Общий вид одного модуля независимого управления двигателем представлен на рисунке 1.18. Рисунок 1.18 представлен ниже.

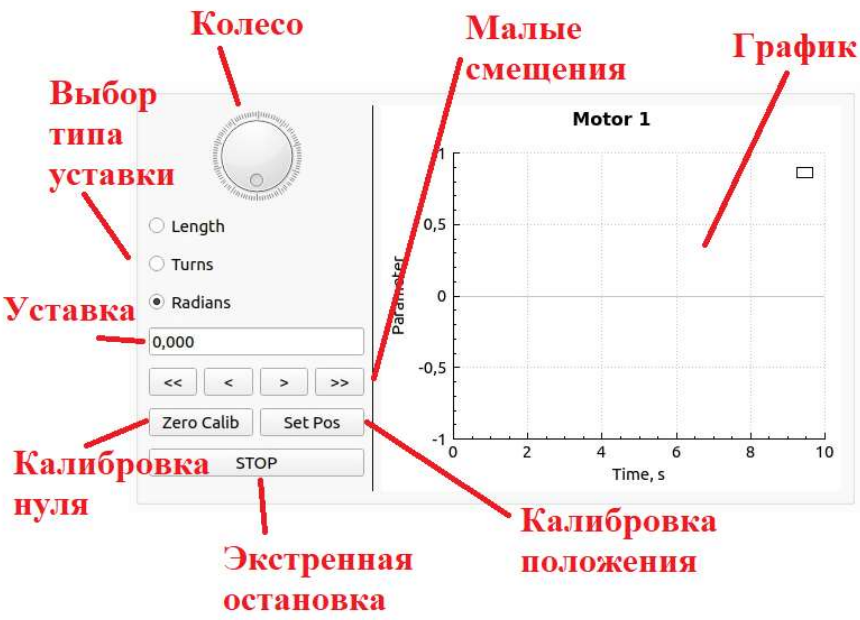


Рисунок 1.18 – Общий вид модуля независимого управления двигателем

Функционал каждого из 4 таких модулей следующий.

С помощью выбора единиц измерения возможно задать формат задаваемых для управления данных. Это могут быть длина выпущенного троса в метрах, количество совершенных выходным валом оборотов, а также количество оборотов в радианах.

Колесом возможно выпустить или подтянуть трос. Количество оборотов обсчитывается согласно выбранным единицам исчисления. Так, если выбрать обороты, то 1 оборот колеса повернет бобину с тросом на один оборот. А если выбрать радианы, то один оборот колеса уже будет соответствовать 6,28 радианам.

В строке задания уставки возможно вручную задать уставку для двигателя. Уставку возможно задавать по длине, оборотам или в радианах.

Кнопки малых смещений позволяют подогнать длину троса без необходимости забивать их каждый раз вручную.

Кнопка калибровки нуля позволяет выставить ноль в качестве текущего положения вала двигателя.

Кнопка калибровки длины позволяет по замеренной длине троса выставить соответствующее положение для двигателя.

Кнопка экстренной остановки останавливает двигатель при нажатии.

Поскольку нет аппаратного средства измерения длины выпущенного троса, его длину необходимо вычитывать косвенно. Для этого используется преобразование из радиан в длину выпущенного троса функцией полинома второго порядка. Для получения данной функции используется следующий подход. Трос сматывается до тех пор, пока карабин не касается планки двигателя. Далее происходит итеративная размотка троса по 6,28 радиан за итерацию. На каждой итерации замеряется длина троса. Размотка заканчивается, когда заканчивается рабочая область троса. Как только набор точек получен, происходит аппроксимация данного набора точек полиномом второго порядка, на основании чего синтезируются коэффициенты полинома. Данный полином интегрирован в программу, что позволяет переводить текущий угол оборота бобины в длину выпущенной части троса. Получившиеся в ходе калибровки полиномы представлены в таблице 1.15. Таблица 1.15 представлена ниже.

Таблица 1.15 – Полиномы двигателей

Номер двигателя	**Калибровочный полином**
Двигатель 1	$l(w) = -0.0002837w^2 + 0.0955636w + 0.1271212$
Двигатель 2	$l(w) = -0.0002618w^2 + 0.0919147w + 0.1280909$
Двигатель 3	$l(w) = -0.0002752w^2 + 0.0963831w + 0.1300303$
Двигатель 4	$l(w) = -0.0002600w^2 + 0.0912814w + 0.1263030$

В данном модуле независимого управления двигателем также присутствуют координатные оси для построения графиков. Для графиков имеется набор минимальных настроек. Настройки графиков представлены в таблице 1.16. Таблица 1.16 представлена ниже.

Таблица 1.16 – Настройки графиков

Название	**Описание**
Is Active	Разрешение построения графиков (по умолчанию разрешено)
Auto Rescale	Задание автоматического масштабирования графика

Rescale to Default	Масштабировать по размеру графиков
Active Registers	Открывает меню выбора активных для построения графиков. Установка «галочки» напротив одного из параметров данного меню означает начало отображения данного параметра на графике
Save Plot	Сохранение изображения графика
Save Data	Сохранение набора данных, отображаемых на графике на текущий момент
Fullscreen	Выделение графика в отдельном окне с возможностью раскрытия на весь экран

В правой части окна располагается область управления пространственным перемещением объекта. Общий вид представлен на рисунках 1.19 и 1.20. Рисунки 1.19 и 1.20 представлены ниже.

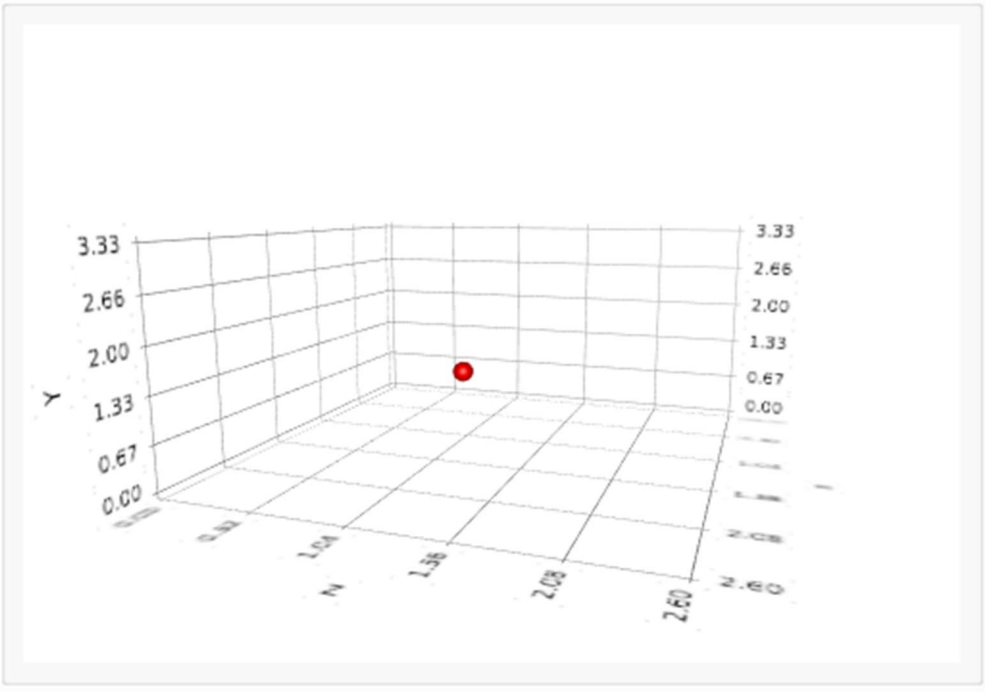


Рисунок 1.19 – График пространственного перемещения объекта

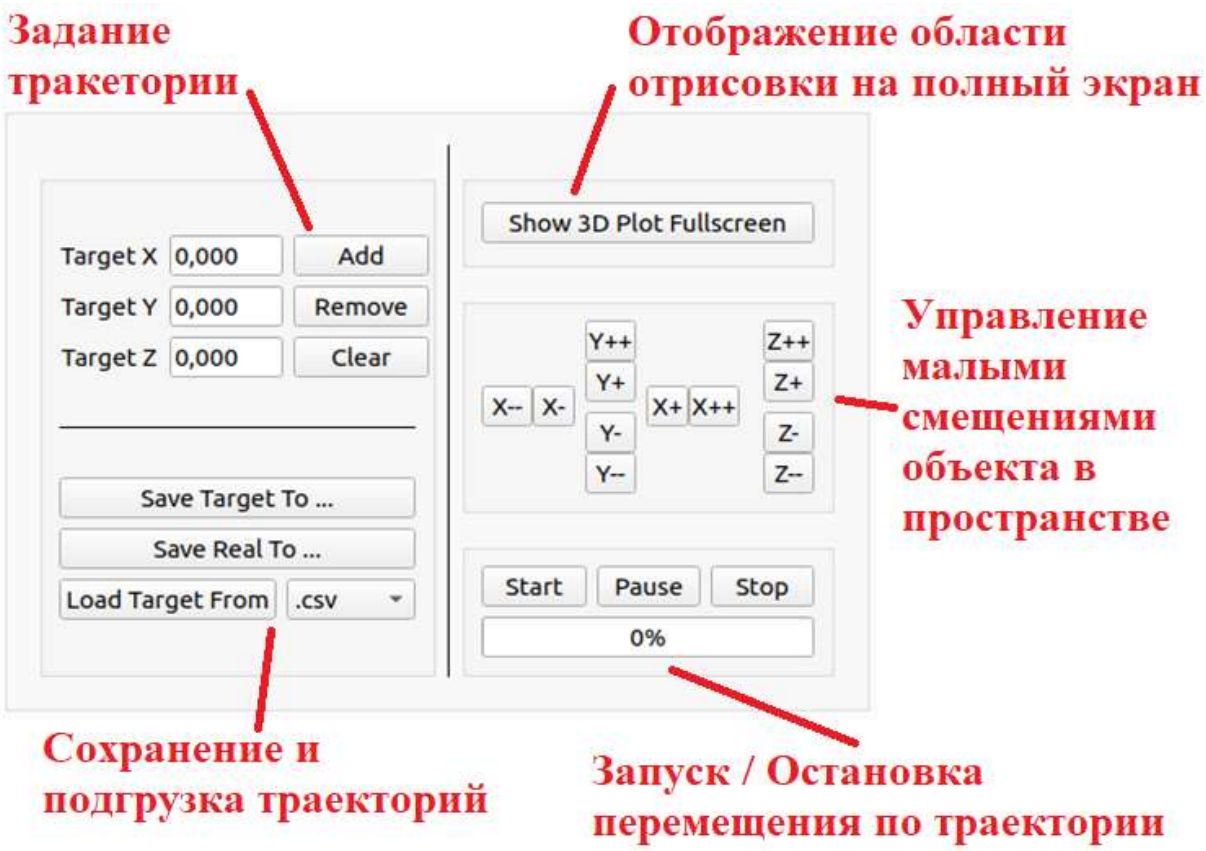


Рисунок 1.20 – Панель задания пространственной траектории для передвижения

График пространственного перемещения отображает текущую позицию объекта, действительную траекторию перемещения, а также заданную траекторию перемещения.

Панель управления позволяет задать траекторию перемещения для объекта. На данной панели имеется область задания траектории, в которой возможно внести простейшую траекторию по точкам. Помимо этого, возможно подгрузить траекторию из внешнего файла, сгенерированного с помощью математического интерпретатора (будет рассмотрена далее). На данной панели также имеется возможность для сохранения действительной и заданной траектории в *.csv файлы. Кнопки малых перемещений позволяют вручную подогнать объект малыми перемещениями вдоль осей X, Y, Z. В нижней части отображается панель прогресса выполнения траектории. В верхней части располагается кнопка вывода трехмерной области в режим отдельного окна.

Получение пространственного положения объекта производится косвенным путем. Для этого, по известным длинам четырех тросов, производится трилатерационное преобразование. Данное преобразование позволяет оценить по длинам тросов реальное положение объекта в пространстве. Алгоритм выполнения трилатерационного преобразования представлен ниже.

Прямое трилатерационное преобразование позволяет получить точку в пространстве из четырех длин тросов.

$$X = \frac{(W^2 - R_3^2 + R_2^2)}{2W},$$

$$Y = \frac{(L^2 - R_1^2 + R_2^2)}{2L},$$

$$Z = H - \sqrt{R_0^2 - (X - W)^2 - (Y - L)^2},$$

где

$R_{0\text{-}3}$ – длины тросов двигателей,

H – высота точек выпуска тросов (все четыре точки располагаются в одной горизонтальной плоскости),

W – расстояние между соседними точками выпуска тросов,

X, Y, Z – конечные координаты объекта.

Обратное трилатерационное преобразование позволяет получить длины тросов по координатам объекта.

$$R_0 = \sqrt{(X - W)^2 + (Y - L)^2 + (Z - H)^2},$$

$$R_1 = \sqrt{X^2 + (Y - L)^2 + (Z - H)^2},$$

$$R_2 = \sqrt{X^2 + Y^2 + (Z - H)^2},$$

$$R_3 = \sqrt{(X - W)^2 + Y^2 + (Z - H)^2}.$$

Общий вид схемы для трилатерационных преобразований представлен на рисунке 1.21. Рисунок 1.21 представлен ниже.

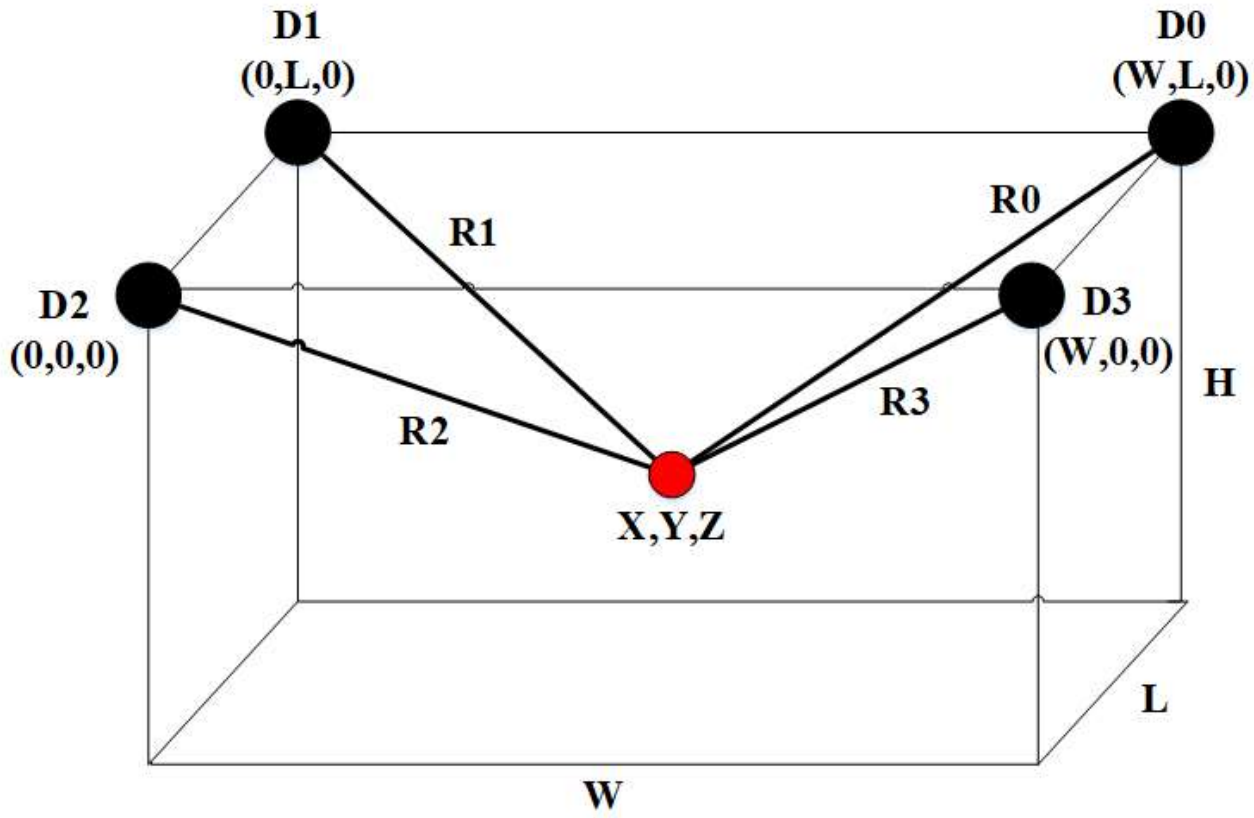


Рисунок 1.21 – Общий вид схемы для трилатерационных преобразований

Таким образом, общая схема преобразований из количества радиан поворота бобин с тросами в пространственное положение объекта включает в себя два преобразования. Первое из них – оценка длин тросов посредством полинома второго порядка, оно позволяет, посредством индивидуально откалиброванных моделей для каждого двигателя, оценить выпущенную длину тросов. Второе преобразование включает в себя трилатерационное преобразование на основании размеров области и длин выпущенных тросов.

Далее была произведена оценка точности оценки положения объекта в пространстве. Максимальная ошибка позиционирования вдоль каждой из осей составляет от 5 до 8 сантиметров, что на общих размерах области функционирования составляет около 1-2 %.

В дальнейшем возможно улучшить позиционирование путем введения оценки положения посредством интеграции в основную программу модуля определения положения ArUco маркера в пространстве (данный модуль описан в дальнейшем).

Общая архитекутра модуля управления представлена на рисунке 1.22. Рисунок 1.22 представлен ниже.

На рисунке 1.22 помеченные аббревиатуры имеют следующую расшифровку.

RLE (Rope Length Estimation) – косвенная оценка длин тросов по показаниям угловых положений бобин с тросом на основе полиномиальной модели.

APE (Angular Position Estimation) – косвенная оценка углов поворота бобины с тросом по показаниям длин тросов на основе полиномиальной модели.

DTT (Direct Trilateration Transform) – прямое трилатерационное преобразование, получение координат объекта в пространстве по длинам тросов.

ITT (Inverse Trilateration Transform) – обратное трилатерационное преобразование, получение длин тросов по положению объекта в пространстве.

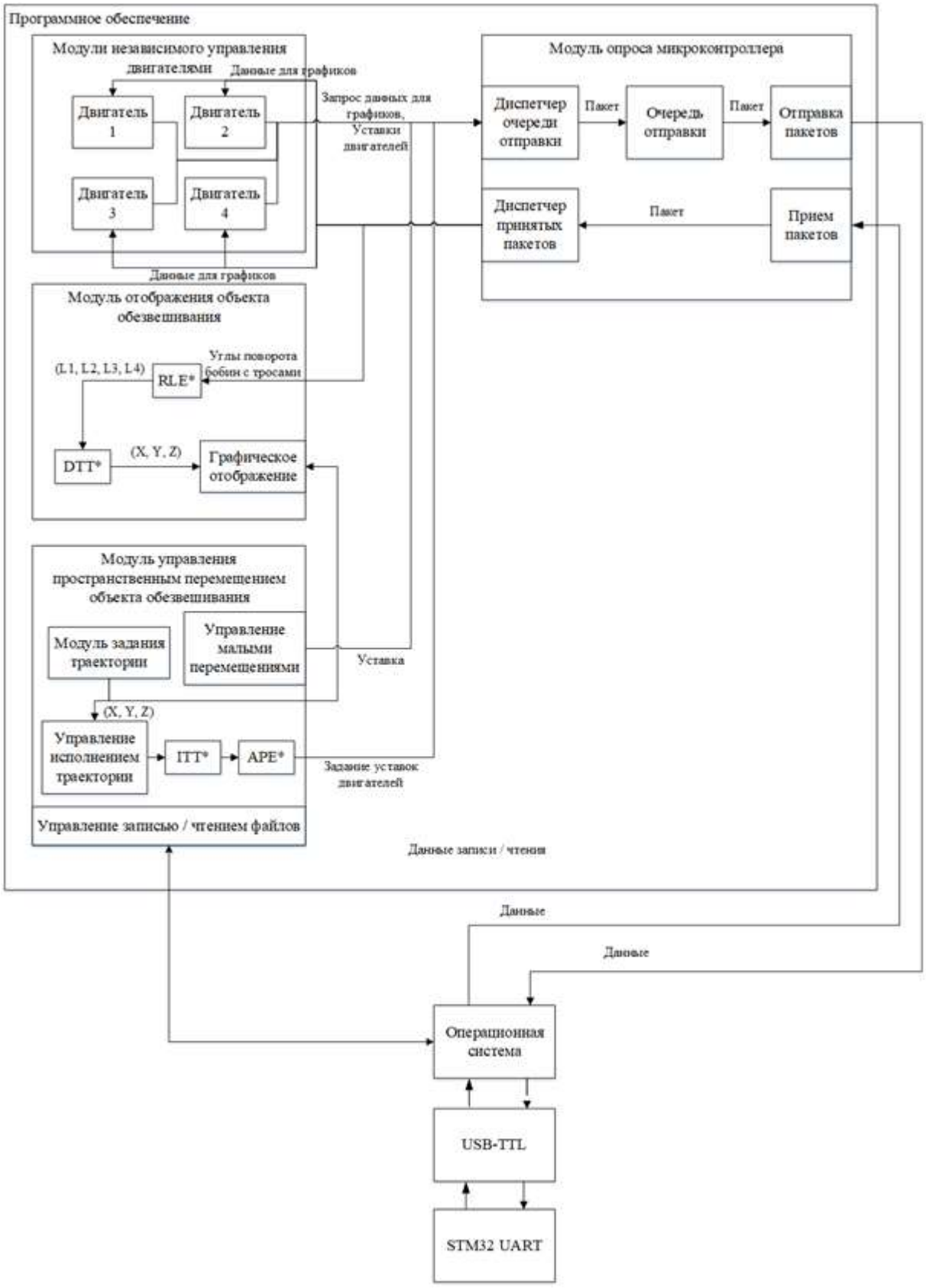


Рисунок 1.22 – Архитектура модуля управления

1.5.4 Экспериментальный модуль блочного программирования

Программное обеспечение для блочного программирования представляет собой совокупность инструментов для построения систем управления тросами при тросовом обезвешивании. Подобные программные обеспечения позволяют ознакомиться с функционалом установки без необходимости использования API и без необходимости реализации нижнего уровня взаимодействия с установкой.

Подобное программное обеспечение имеется у многих продуктов, в частности у FESTO для их мобильной платформы FESTO Robotino. Общий вид среды блочного программирования для FESTO Robotino [9] представлен на рисунке 1.23.

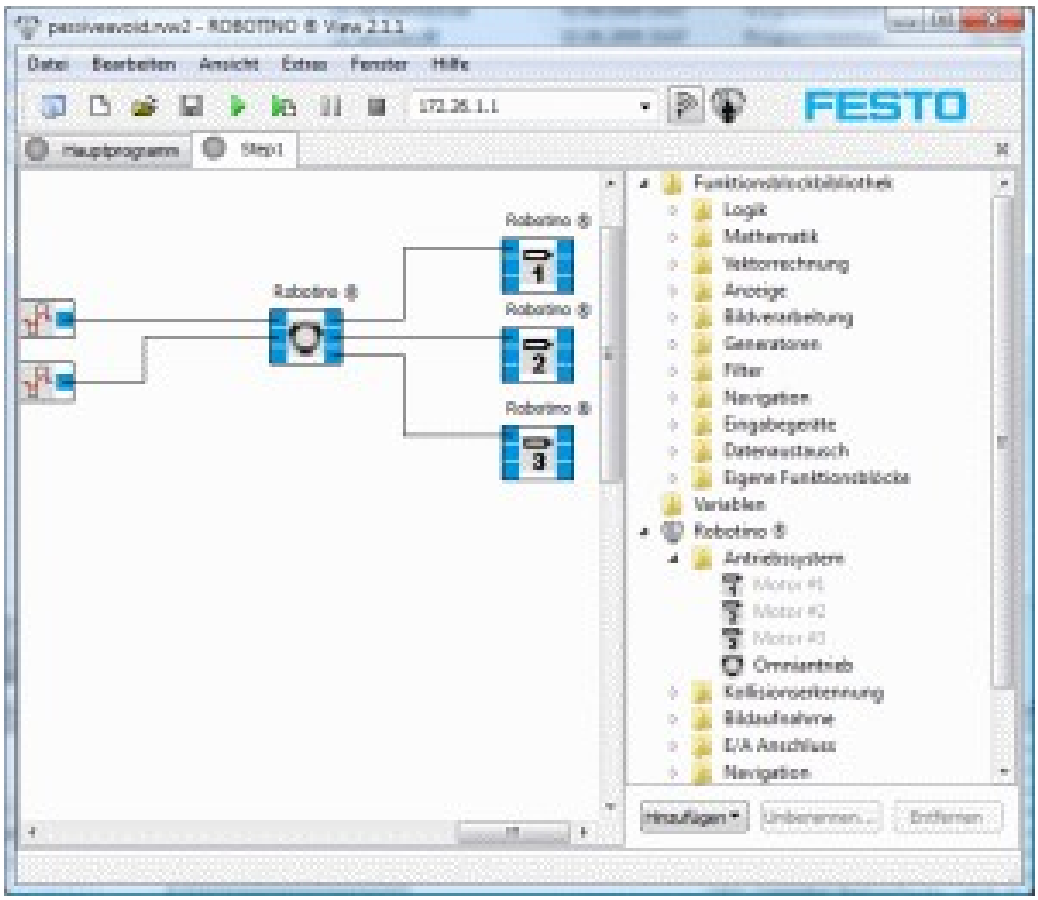


Рисунок 1.23 – Среда блочного программирования для FESTO Robotino

Для установки тросового обезвешивания подобная среда будет иметь схожее предназначение. Она позволит реализовать алгоритмы собственные алгоритмы без необходимости программирования на каких-либо языках.

Данная среда блочного программирования получила название Essential Block Application Layout Assembler. Она строится на основании взаимодействия блоков. Общий вид интерфейса на текущий момент представлен на рисунке 1.24. Рисунок 1.24 представлен ниже.

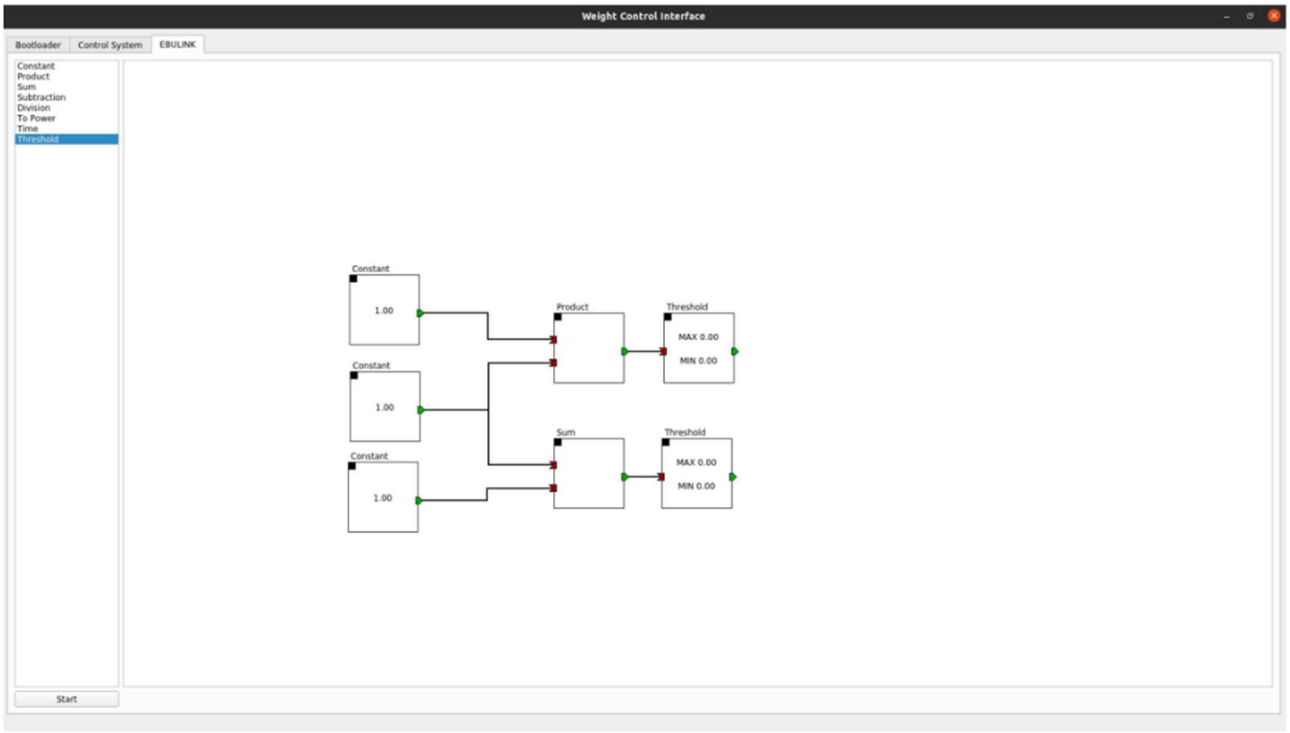


Рисунок 1.24 – Общий вид разрабатываемого раздела блочного программирования

Изображение типового блока представлено на рисунке 1.25. Рисунок 1.25 представлен ниже.

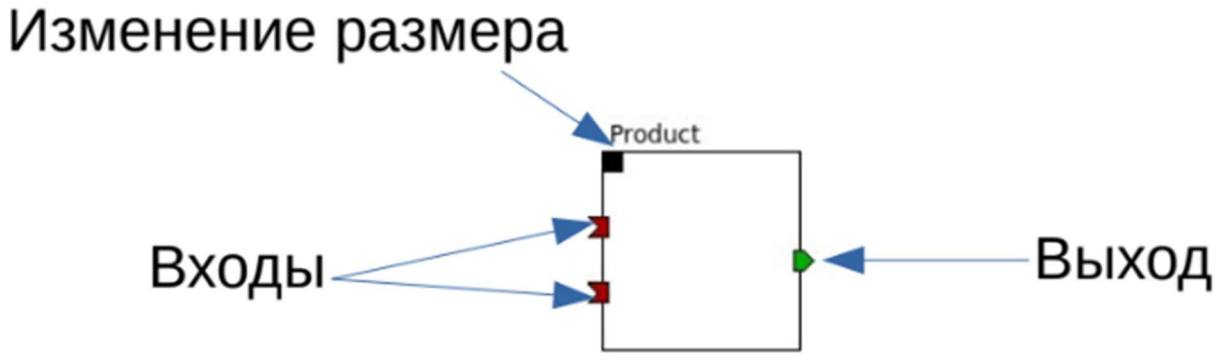


Рисунок 1.25 – Типовой блок

В верхнем левом углу блока имеется точка, потянув за которую возможно изменение размера блока. В левой части блока располагаются входы

для блока, на которые возможно подавать сигналы. В правой части блока располагаются выходы, сигнал из которых можно завести на один из входов другого блока. При наведении курсора на вход или выход показывается подсказка, уведомляющая пользователя о назначении выхода. Название блока указывается в верхней его части, непосредственно над блоком. Некоторые блоки могут содержать текст непосредственно в себе, это необходимо для возможности непосредственного отображения данных (например, константы, ограничителя и т.д.).

Некоторые блоки имеют в себе параметры. Таковыми блоками являются блоки ПИД-регуляторов, ограничителей, констант и т.д. Для изменения параметра блока необходимо кликнуть дважды по блоку. После этого всплывает меню, в котором возможно задание значения параметра. Данная операция наглядно представлена на рисунке 1.26. Рисунок 1.26 представлен ниже.

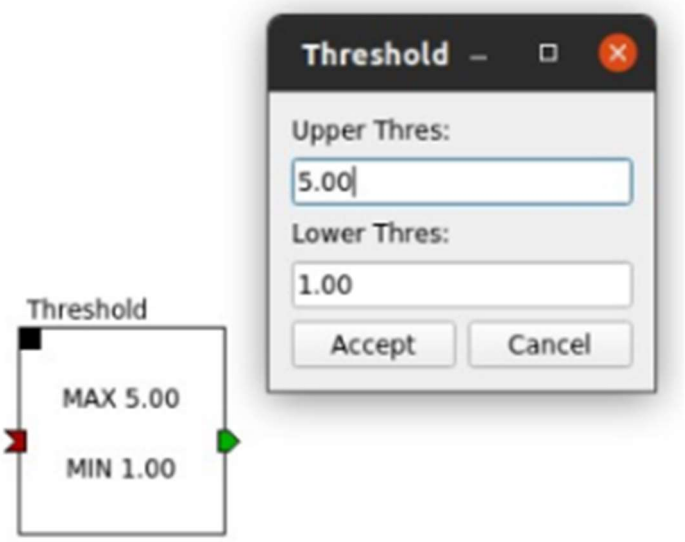


Рисунок 1.26 – Операция задания параметра блока ограничителя

Одним из последних нововведений в системе стал блок графического отображения информации. Данный блок позволяет выводить обрабатываемую информацию на график, после чего сохранять изображение графика и данные. Общий вид блока графической обработки информации представлен на рисунке 1.27.

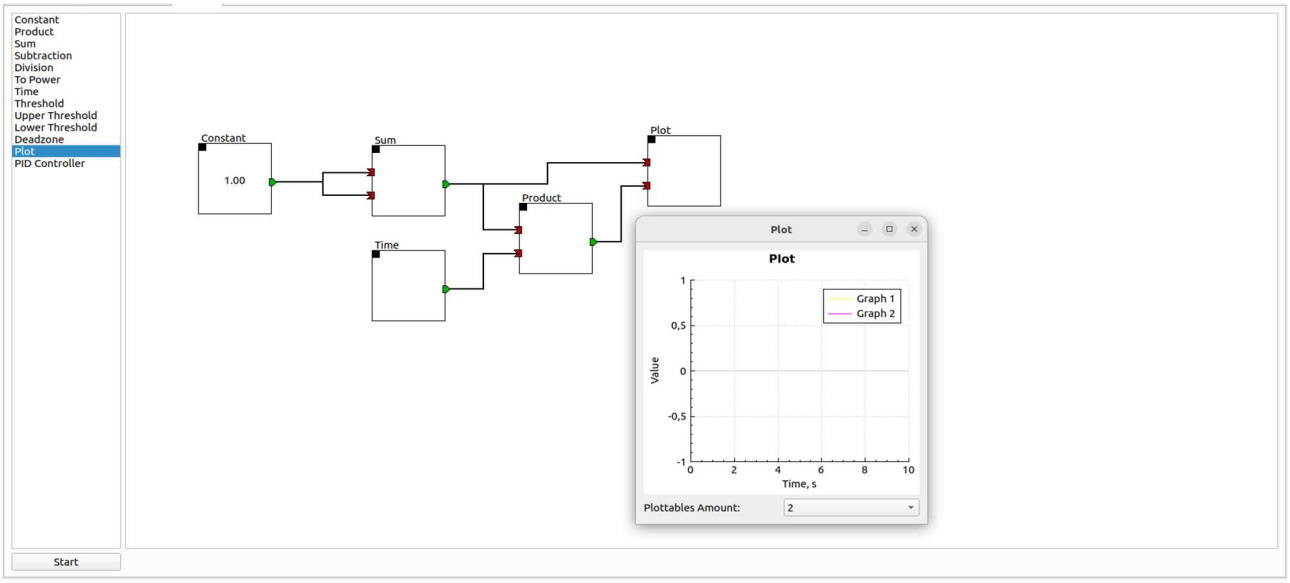


Рисунок 1.27 – Блок графической обработки информации

Для соединения блоков в среде используются сигнальные линии. Сигнальные линии начинаются от сигнала какого-либо блока и заканчиваются на слоте какого-либо блока. По сигнальным линиям, на текущий момент идут сигналы с данными одного типа – float. Это было сделано для упрощения разработки на начальном этапе. В дальнейшем предусматривается введение других типов сигналов. Введение других типов сигналов необходимо для корректной обработки булевых значений, а также тех значений, которые представлены в бинарном виде. Поскольку на текущий момент тип сигнала всего один, то сигнал любого блока может быть идти в слот любого блока.

На текущий момент в систему были добавлены блоки, представленные в таблице 1.17.

Таблица 1.17 – Имеющиеся блоки для блочного программирования

Название блока	Описание блока
Constant	Блок константы
Product	Блок математической операции «Произведение»
Sum	Блок математической операции «Сложение»
Subtraction	Блок математической операции «Вычитание»
Division	Блок математической операции «Деление»

To Power	Блок математической операции «Возведение в степень»
Time	Блок глобального времени в мс
Threshold	Блок ограничителя сверху и снизу
Upper Threshold	Блок ограничителя сверху
Lower Threshold	Блок ограничителя снизу
PID	Блок ПИД-регулятора
Plot	Блок графического отображения информации
Derivative	Блок взятия производной
Integrator	Блок интегрирования

После завершения структурной схемы в редакторе необходимо нажать кнопку Start, после чего начнется обработка заданной системы. Данная обработка происходит посредством ядра обработчика.

В отличии от MATlab Simulink, где обработка модели происходит посредством генерирования программного кода, в данном случае схема гораздо проще. После нажатия клавиши Start происходит обработка системы по шагам. За обработку системы отвечает ядро. Функционирование ядра устроено следующим образом.

При запуске системы происходит запуск обработчика, который на каждой итерации выполняет следующий алгоритм обработки. Ядро просматривает выходы блоков с прошлой итерации и заносит эти показания в связанные с этими выходами входы. Если выход на прошлой итерации не был рассчитан (нулевая итерация), то происходит присвоение нуля данному входу. Далее, когда все входы были заполнены значениями сигналов с прошлой итерации, происходит обработка каждого блока с использованием полученных значений входов. После этого выходы обновляются и итерация заканчивается.

Данный способ обработки позволяет очень легко обойти вопрос зацикливания системы (например, при построении RS-триггера). Так как в этом случае обработка на основании показаний может привести к неразрешимости за-

дачи. Однако, такой подход содержит большой недостаток. Данный недостаток заключается в большой инерционности системы и линейно зависит от глубины полученной системы. Таким образом, если система состоит из трех блоков, то последний блок получит данные от первого с задержкой в две итерации. Данная проблема, по своей сути, не является серьезной проблемой. Одним из ее решений может служить отказ от проведения единичных итераций для совершения вычислительного шага. Предлагается использовать вычислительный шаг, который будет включать в себя множество итерационных шагов, причем число этих шагов будет варьироваться, в зависимости от глубины дерева системы регулирования. Итерации внутри вычислительного шага будут выполняться последовательно и подряд, с минимальными промежутками времени между друг-другом, в то время как величина времени вычислительного шага будет значительно больше, что позволит пренебречь числом итераций и практически убрать инерционность из реализованной системы.

Общая архитектура каждого из блоков представлена на рисунке 1.28. Рисунок 1.28 представлен ниже.

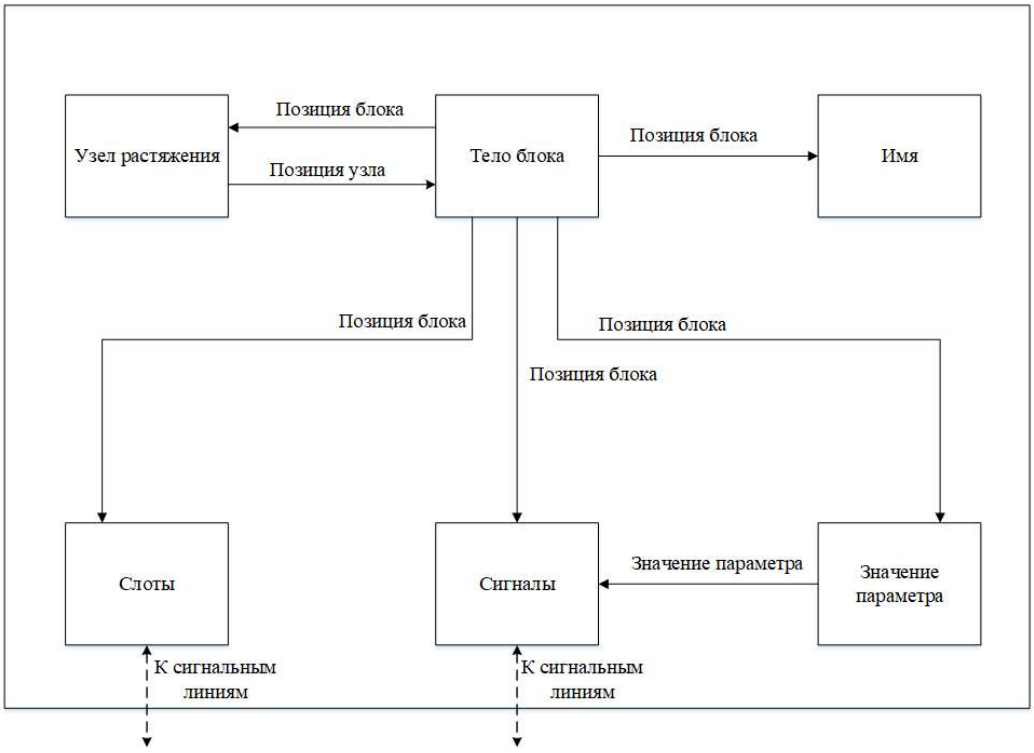


Рисунок 1.28 – Общая архитектура блока

Архитектура взаимодействия блоков представлена на рисунке 1.29. Рисунок 1.29 представлен ниже.

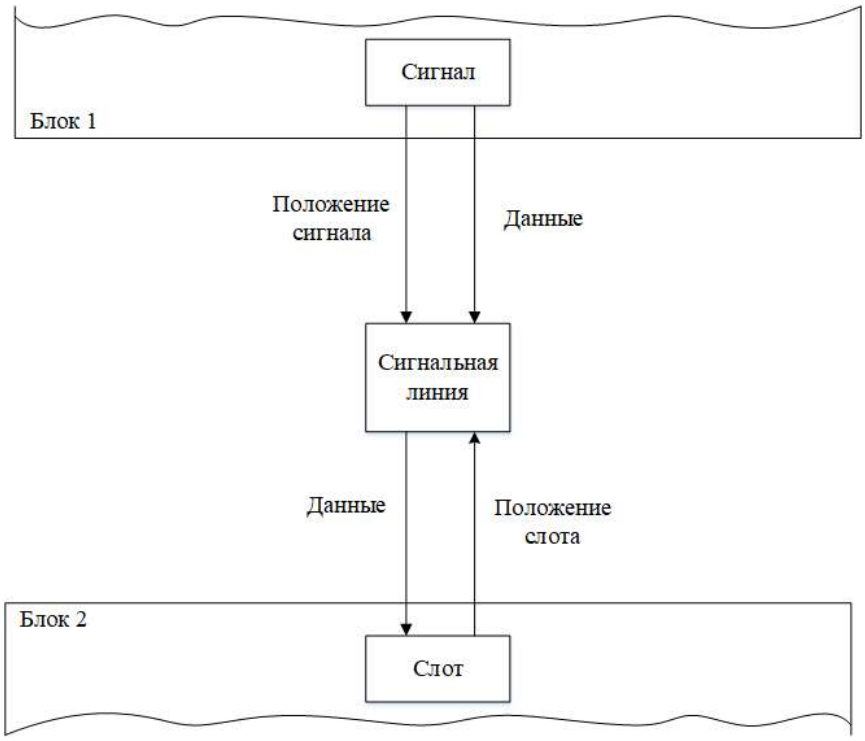


Рисунок 1.29 – Общая архитектура взаимодействия блоков

Данный модуль блочного программирования представлен в качестве экспериментального модуля. На текущий момент он не имеет реализации на базе микроконтроллера.

Более подробно ознакомиться со всем исходным кодом основного ПО на ПК возможно по ссылке https://github.com/eai13/weight_control_system/tree/main_1/WeightControlInterface/WeightControlInterface или по QR-коду

1.5.5 Вспомогательное программное обеспечение

Вспомогательное программное обеспечение включает в себя набор ПО, призванный упростить взаимодействие с основным программным обеспечением для пользователя.

Вспомогательное программное обеспечение включает в себя два модуля.

Первым из них является вспомогательная программа для определения координат обезвешиваемого объекта по размещенному на нем ArUco маркеру.

Вторым вспомогательным программным обеспечением является математический интерпретатор с графической оболочкой для возможности быстрой подстройки графиков и экспорта в нативный формат файла для основного ПО.

1.5.6 Программное обеспечение для определения координат обезвешиваемого объекта через ArUco маркер

Данное вспомогательное программное обеспечение является ПО для получения координат обезвешиваемого объекта прямым способом. Поскольку в основном программном обеспечении положение объекта определяется путем математических преобразований, что является косвенным определением положения, это потребует перекалибровки установки при изменении размеров комнаты или диаметра бобин и само по себе может иметь интегральную ошибку на дистанции. Для возможности более точного позиционирования без накопительной ошибки было разработано программное обеспечение детектирования положения объекта через расположенный на нем ArUco макер.

Общий вид данного программного обеспечения представлен на рисунке 1.30. Рисунок 1.30 представлен ниже.

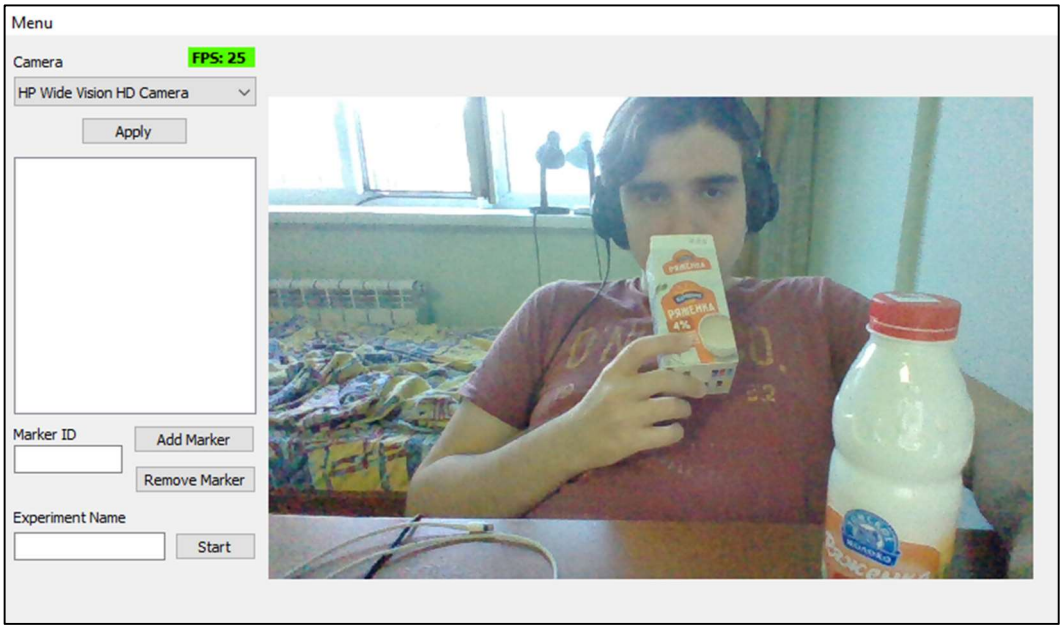


Рисунок 1.30 – Общий вид ПО

В данном ПО возможно добавлять различные ArUco метки в качестве активных по их ID.

В программе производится выбор камеры для подключения, после чего заносятся некоторые ID маркеров, которые при занесении начинают отображаться в списке активных маркеров. Камера имеет ряд настроек, которые осуществляются посредством вызова меню. Настройки камеры показаны на рисунке 1.31. Рисунок 1.31 представлен ниже.

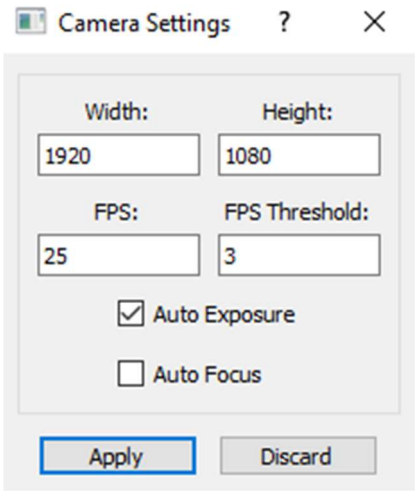


Рисунок 1.31 – Меню настройки камеры

Для отслеживания качества связи в верхней части окна отображено количество кадров в секунду (FPS).

При необходимости записи в файл и сохранения эксперимента необходимо нажать кнопку Start и указать название файла, куда будет происходить запись.

При начале эксперимента в кадре показываются текущее время, номер кадра и отображаются активные маркеры, которые были определены. Общий вид кадра во время эксперимента представлен на рисунке 1.32. Рисунок 1.32 представлен ниже.

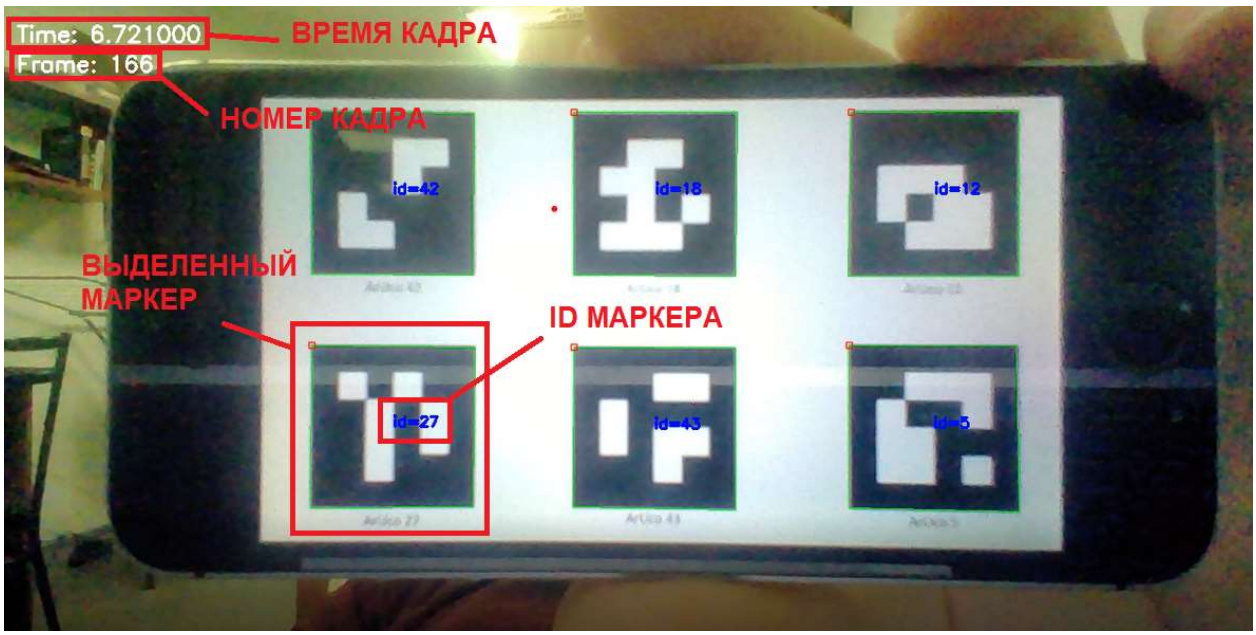


Рисунок 1.32 – Пример кадра во время проведения эксперимента

В результате записи эксперимента производится запись положений углов каждого из маркеров и время. По этим данным затем возможно построить карту передвижения маркера. Пример траектории представлен на рисунке 1.33. Рисунок 1.33 представлен ниже.

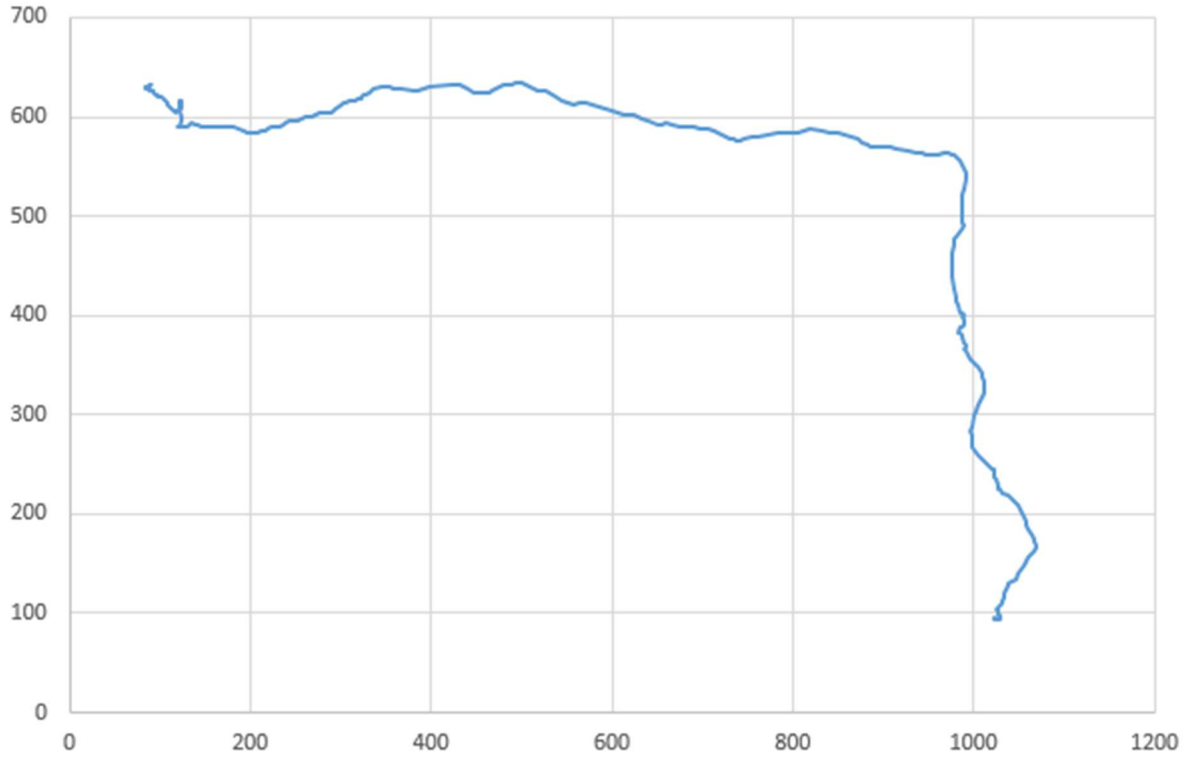


Рисунок 1.33 – Пример траектории

В дальнейшем планируется интеграция данного программного обеспечения непосредственно в интерфейс основного ПО или реализация непосредственного интерфейса для передачи данных между двумя ПО через shared-memory или TCP(localhost).

Программный код проекта представлен по ссылке https://github.com/eai13/aruco_weightlessness_detector или по QR-коду

1.5.7 Математический интерпретатор с графической оболочкой

Поскольку иногда приходится строить достаточно сложные траектории для передвижения объекта в пространстве, становится необходимостью задавать траекторию не через точки, а посредством математических формул. В основном программном обеспечении, как было выше упомянуто, имеется средство для задания траектории либо через отдельные точки (что достаточно трудоемко при задании даже самой примитивной окружности), либо посредством подгрузки файла траектории, который имеет определенный формат.

В связи с этим, при необходимости задания сложной траектории передвижения необходимо каким-либо образом ее задавать в сторонних ПО (Excel, MATlab, Python, C++). Тут возникает вопрос необходимости «изобретения» средств интерпретации программного кода. С одной стороны уже имеются средства для генерации траекторий, но во-первых, файл траектории имеет, хоть и далеко не сложный, но специфичный формат файла, что создает проблемы пользователю при генерации траектории и может ввести его в ступор. С другой стороны, не каждый пользователь имеет на своем ПК Excel, Python и уж тем более обработчик C++. В связи с этим было предложено реализовать свой узкоспециализированный интерпретатор, в который встроить необходимые функции и типы данных для работы и убрать все средства, которые не

имеет смысла оставлять, после чего внедрить его в графическую оболочку и встроить в основное программное обеспечение.

Математический интерпретатор представляет собой набор алгоритмов для работы с пользовательским вводом формата «строка». Примером такого ввода может служить строка вида «x = 1 + 2».

В основе интерпретатора лежит модульное ядро обработчика. Модули являются последовательными и в совокупности позволяют обработать строку и работать с созданными пользователем переменными.

Ядро делится на следующие составляющие: препроцессор строки, интерпретатор строки, препроцессор дерева, интерпретатор дерева. Рассмотрим каждый из модулей ядра в отдельности.

Первым составляющим является препроцессор строки. Препроцессор строки выполняет задачу форматирования строки и передача отформатированной строки на дальнейшую работу интерпретатору строки. Тут происходит проверка правильности расстановки скобок, а также разбитие всей строки на примитивы. Примитивами в данном случае могут являться скобки, числа, имена функций и переменных, операторы и т. д. Также, для возможности работы с отрицательными числами препроцессор строки форматирует производит замену оператора отрицания перед соответствующим числами и переменными на строку «(-1) *». Примером работы препроцессора строки может служить следующее преобразование пользовательской строки «y = -SIN(1 + 2)» в набор примитивов “y”, “=”, “-1”, “*”, “SIN”, “(“, “1”, “+”, “2”, “)”.

Вторым модулем ядра является интерпретатор строки. Интерпретатор строки берет набор примитивов, которые получились после обработки препроцессором строки. Интерпретатор строки выполняет задачу сортировки примитивов в необходимом порядке для дальнейшего формирования дерева препроцессором дерева. На данном этапе вводятся понятия функции и операторов. Функции и операторы имеют количественный приоритет исполнения (подобно тому, как операция умножения имеет более высокий приоритет исполнения по сравнению с оператором сложения в классической математике). При

этом, если имеется две одинаково приоритетных операции, первой исполнится та, которая стояла в строке левее (правее для арабов). Операторы распределяются по приоритетам так же, как и в классической математике, при этом все функции имеют одинаковый приоритет, который выше чем у всех операторов. Благодаря введению рекурсивного прогона примитивов в приоритетности не приходится учитывать скобки, они учитываются автоматически (про обработку скобок будет описано далее). Общая сводная таблица приоритетов представлена в таблице 1.18. Таблица 1.18 представлена ниже.

Процесс выполнения интерпретации строки строится следующим образом. Сперва создается стек для хранения операторов и функций., а также создается список, куда постепенно заносятся примитивы в нужном порядке. Далее, начиная с первого примитива в исходном списке, полученном из препроцессора строки, происходит следующее. Если примитив представляет собой переменную или число, то он заносится в выходной список примитивов. Если встречается оператор или функция, происходит занесение этой операции на стек. При этом, если на верхушке стека лежит такая же или большая по приоритету операция, то происходит поэлементный перенос примитивов операций из стека в выходной список до тех пор, пока не встречается менее приоритетная операция. Как только такая операция была замечена, происходит остановка раскрутки стека и текущая операция складывается наверх стека.

Таблица 1.18 – Приоритеты операций

Название операции	Количественный приоритет
Присвоение (=)	0x00
Сложение (+), Вычитание (-)	0x01
Умножение (*), Деление (/)	0x02
Возведение в степень (^)	0x03
Все функции	0xFF

Если на пути интерпретатора строки встречается открывающаяся скобка, то для абстракции и упрощения работы кода, происходит поиск соответствующей ей по типу и глубине скобки и для всего того, что было внутри

скобки происходит рекурсивный вызов интерпретатора строки исключительно для этого набора примитивов. Как только рекурсивно вызванный интерпретатор строки завершил работу он возвращает отформатированный набор примитивов, который без изменений заносится в выходной список примитивов.

Таким образом, после интерпретатора строки получается выходной набор примитивов, который далее передается на построение дерева препроцессору дерева. Пример совместной работы препроцессора и интерпретатора строки представлен в таблице 1.19. Таблица 1.19 представлена ниже.

Таблица 1.19 – Результат совместной работы препроцессора и интерпретатора строки

Стадия	**Значение**
Исходная строка	«y = -SIN(1 + 2)»
После препроцессора строки	«y», «=», «-1», «*», «SIN», «(», «1», «+», «2», «)»
После интерпретатора строки	«y», «-1», «1», «2», «+», «SIN», «*», «=»

Общий алгоритм совместной работы препроцессора и интерпретатора строки представлен на рисунке 1.34. Рисунок 1.34 представлен ниже.

После интерпретатора строки в дело вступает препроцессор дерева. Основной задачей препроцессора дерева является построение готового дерева для работы интерпретатора дерева по исходному упорядоченному набору примитивов, полученному от интерпретатора строки.

Создаваемый язык для интерпретатора создается максимально простым, в связи с чем не имеет строгой типизации. В связи с этим одной из задач препроцессора строки также является проверка типов данных для всех функций, операторов, переменных и значений, а также задание типов данных для новых переменных, которые создаются пользователем.

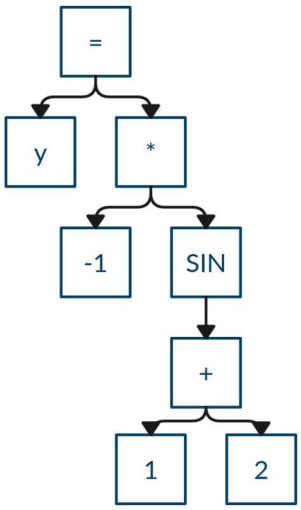
Само конечное дерево имеет следующий вид. В корне дерева лежит самая низкоприоритетная операция. Листьями дерева являются значения и пере-

менные. Каждый узел дерева заполняется элементами справа налево. Количество связей для каждого узла дерева свое и зависит от того, какое количество аргументов может принимать оператор или функция. Все узлы являются функциями или операторами.

Общий алгоритм построения дерева достаточно прост. Берется список примитивов, полученный от интерпретатора строки, по нему выполняется итеративный проход из конца в начало (справа налево). Все примитивы добавляются в самое правое свободное плечо самого правого узла дерева. У узлов функций это число задается при создании функции, у узлов операторов это количество всегда равняется двум. По ходу заполнения проверяется правильность типов аргументов, а также тем переменным, которые только создаются и не имеют типа данных присваивается тип данных исходя из того, какой тип данных вернет создающий ее оператор присваивания.

Пример работы препроцессора дерева представлен в таблице 1.20. Таблица 1.20 представлена ниже.

Таблица 1.20 – Пример работы препроцессора дерева

Стадия	**Значение**
После интерпретатора строки	«y», «-1», «1», «2», «+», «SIN», «*», «=»
После препроцессора дерева	

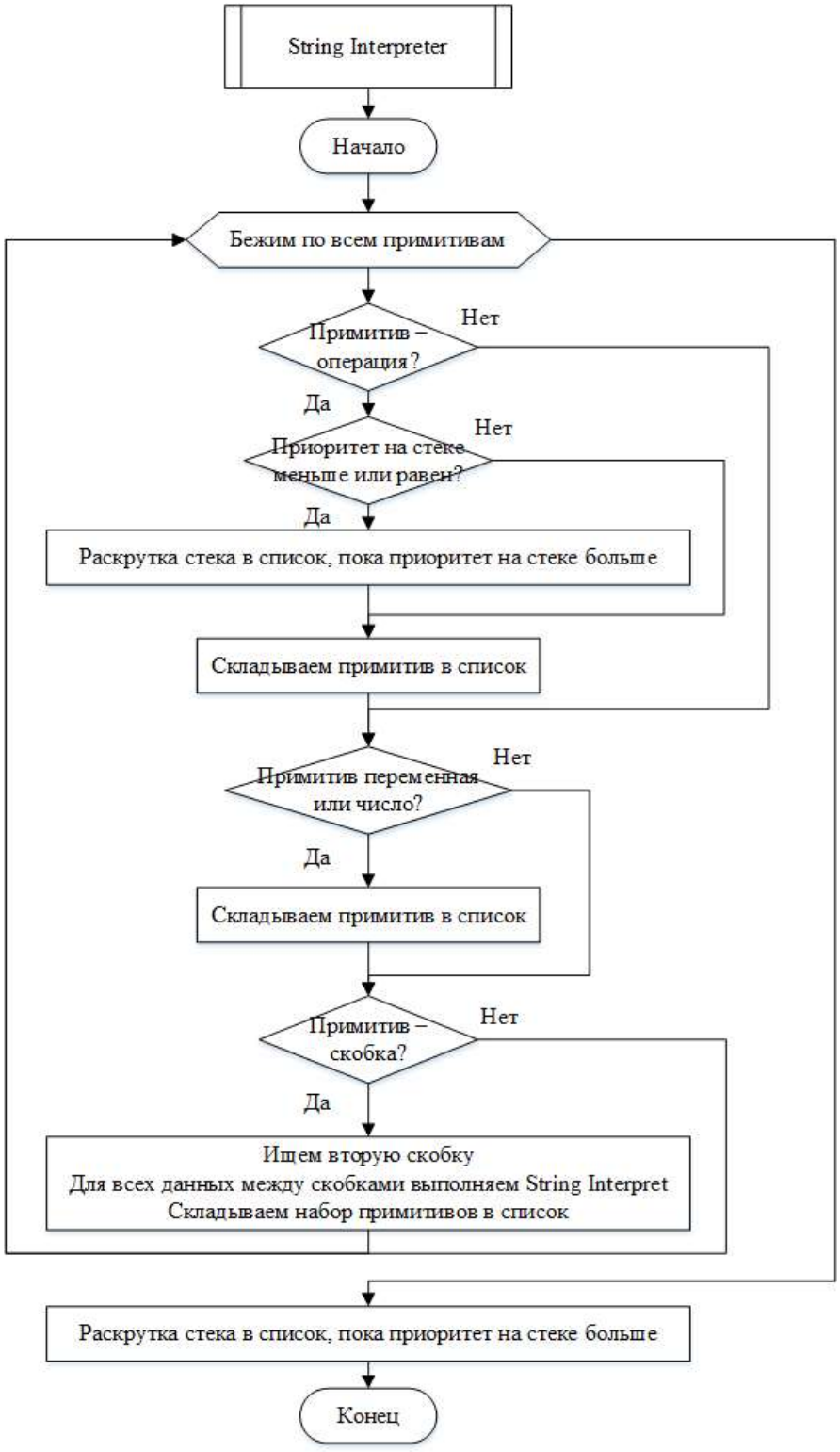


Рисунок 1.34 – Блок-схема препроцессора и интерпретатора строки

Завершающим этапом работы всего интерпретатора является интерпретатор дерева. Данный интерпретатор дерева непосредственно производит все расчеты по дереву, предоставленному препроцессором дерева. Расчеты произ-

водятся рекурсивным способом, начинаясь в корне дерева и расходясь до листьев. Его работа крайне проста и не нуждается в большем описании. В вышеописанном примере интерпретации пользовательской строки результатом станет создание новой переменной «y» со значением, равным «-1 * SIN(1 + 2)».

При реализации интерпретатора была заложена следующая архитектура. Основной задачей была разработка архитектуры, поддерживающей достаточно простую интеграцию новых типов данных, а также новых функций и операторов. В качестве минимально необходимых типов были введены типы DOUBLE и VECTOR. На уровне исходного кода программы возможно добавлять новые функции и типы данных, просто описав их взаимодействие.

Поскольку использование интерпретатора из консоли является не самым удобным вариантом для пользователя, было решено интегрировать его в графическую оболочку. Графическая оболочка выполнялась по образу и подобию программы MATlab. Общий вид графической оболочки представлен на рисунке 1.35. Рисунок 1.35 представлен ниже.

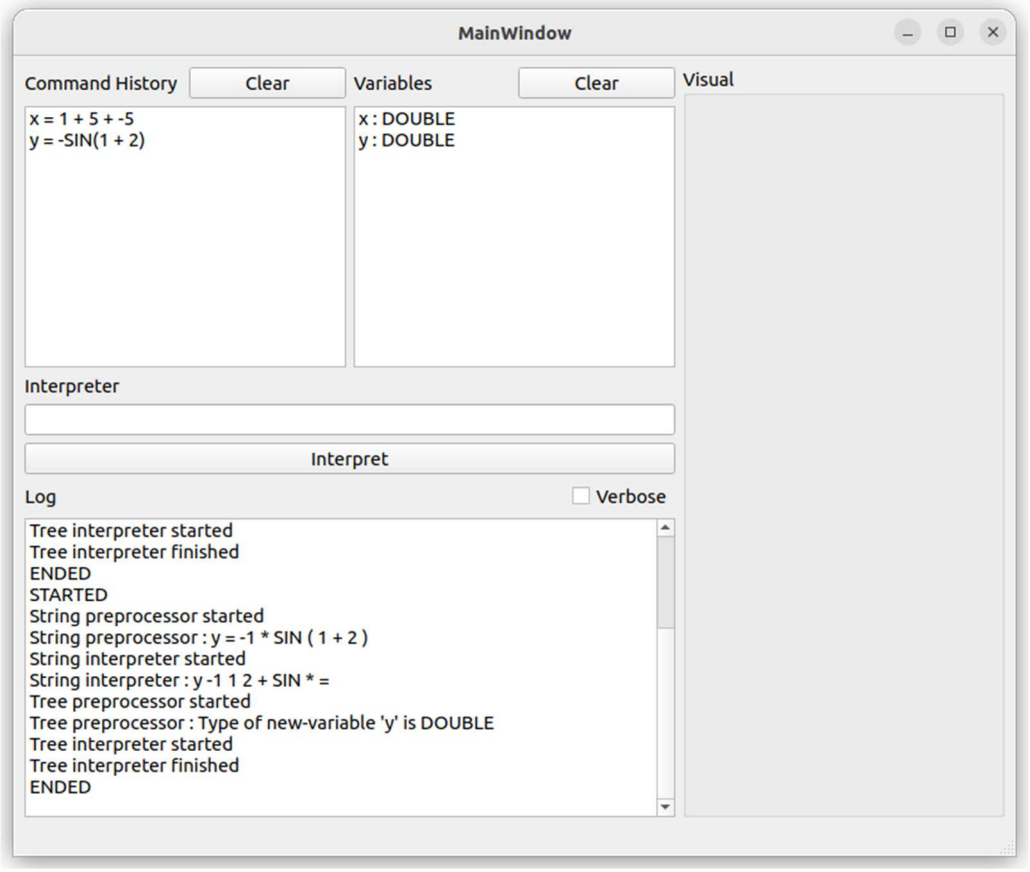


Рисунок 1.35 – Общий вид графической оболочки интерпретатора

В верхней представлена история команд, из которой возможно повторить исполненную ранее команду. Также в верхней части отображаются актуальные переменные, которые были созданы пользователем. Двойным нажатием по имени переменной возможно вызвать графическое меню настройки переменной. Для типа VECTOR это будет таблица, для типа DOUBLE это будет одно поле для ввода данных.

В средней области располагается строка для ввода интерпретируемой строки пользователя. После нажатия кнопки Interpret строка интерпретируется и данные о статусе интерпретации заносятся в нижнюю область интерфейса «Log».

Наряду с базовыми операторами в данном интерпретаторе имеется обработка следующих функций (таблица 1.21).

Таблица 1.21 – Описание функций интерпретатора

Функция	**Описание**
SIN	Синус
COS	Косинус
TAN	Тангенс
CTG	Котангенс
ABS	Модуль
EXP	Функция возведения экспоненты в степень
LOG	Функция 10-чного логарифма
RANGE	Создание массива в формате от-до-шаг
PLOT2D	Построение двухмерного графика (рис. 3)
PLOT3D	Построение трехмерного графика (рис. 4)
EXPORT3DTRAJ	Экспортирование 3D траектории в формате, подходящем для импорта в основном приложении управления системой обезвешивания (рис. 5)

Примеры использования функций PLOT2D и PLOT3D представлены на рисунках 1.36 и 1.37. Рисунки 1.36 и 1.37 представлены ниже.

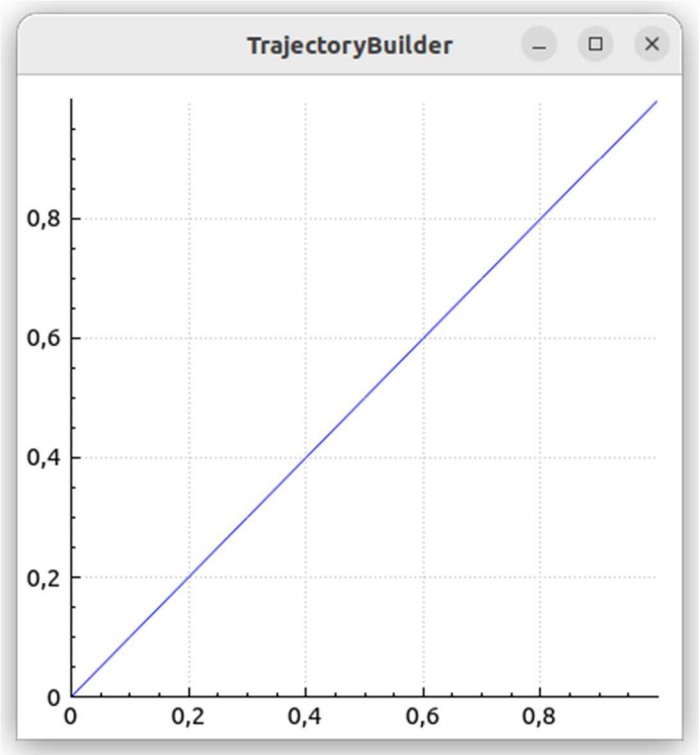


Рисунок 1.36 – Пример построения 2D траектории

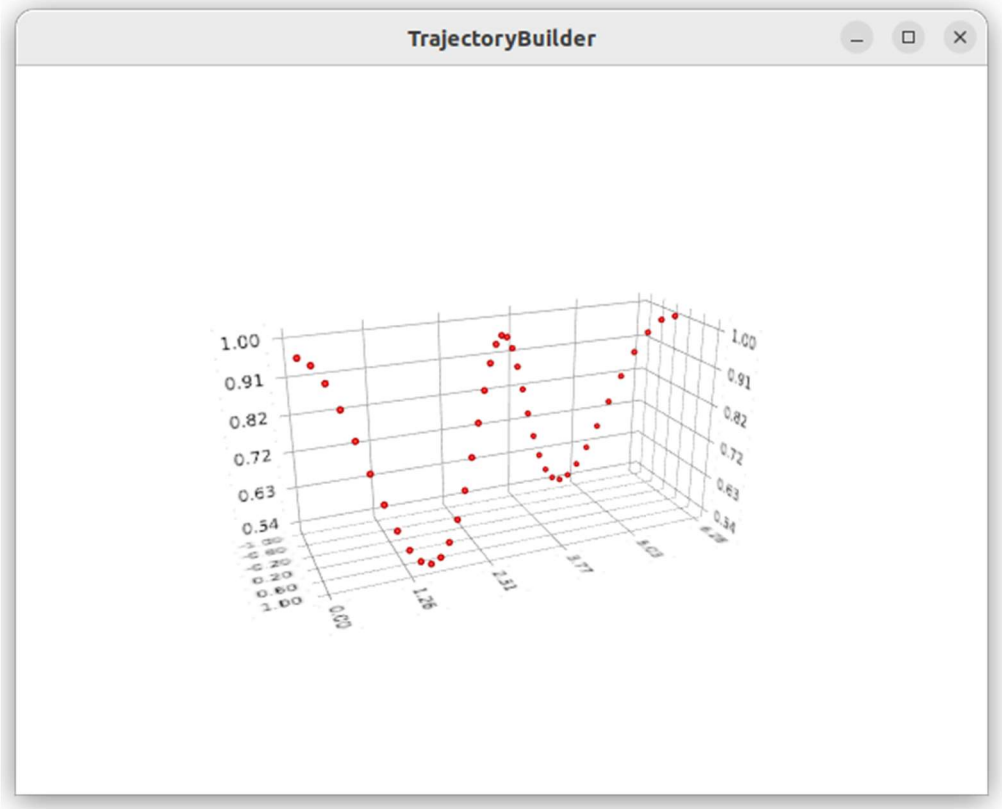


Рисунок 1.37 – Пример построения 3D траектории

Экспорт траектории происходит в csv-файл в специфичном формате. В таком же формате на вход принимается уставочная траектория в основном ПО.

Пример формата csv-файла уставочной траектории представлен на рисунке 24. Рисунок 1.38 представлен ниже.

	A	B	C
1	X	Y	Z
2	1	2	1
3	1.31	1.95	1
4	1.59	1.81	1
5	1.81	1.59	1
6	1.95	1.31	1
7	2	1	1
8	1.95	0.69	1
9	1.81	0.41	1

Рисунок 1.38 – Пример сформированного файла траектории

Таким образом, с помощью данного интерпретатора и графического интерфейса для него возможно генерировать сложные траектории для управления системой обезвешивания.

Программное обеспечение интерпретатора и графической оболочки для него находится на стадии доработки. Ознакомить с ним и внести предложения по разработке возможно по ссылке https://github.com/eai13/weight_control_system/tree/main_1/TrajectoryBuilder/TrajectoryBuilder или по QR-коду

1.6 Экспериментальный пуск установки

В качестве эксперимента пусковой установки было решено выполнить траекторию при перемещении объекта обезвешивания по окружности в пространстве.

Для этого была выбрана горизонтальная плоскость. Общий вид заданной траектории представлен на рисунке 1.39. Рисунок 1.39 представлен ниже.

Рисунок 1.39 – Уставочная траектория для передвижения

По заданной траектории было запущено управление объектом обезвешивания через основное ПО.

В результате обезвешивания была успешно выполнена траектория. Объемный вид траектории можно видеть на рисунке 1.40. Рисунок 1.40 представлен ниже.

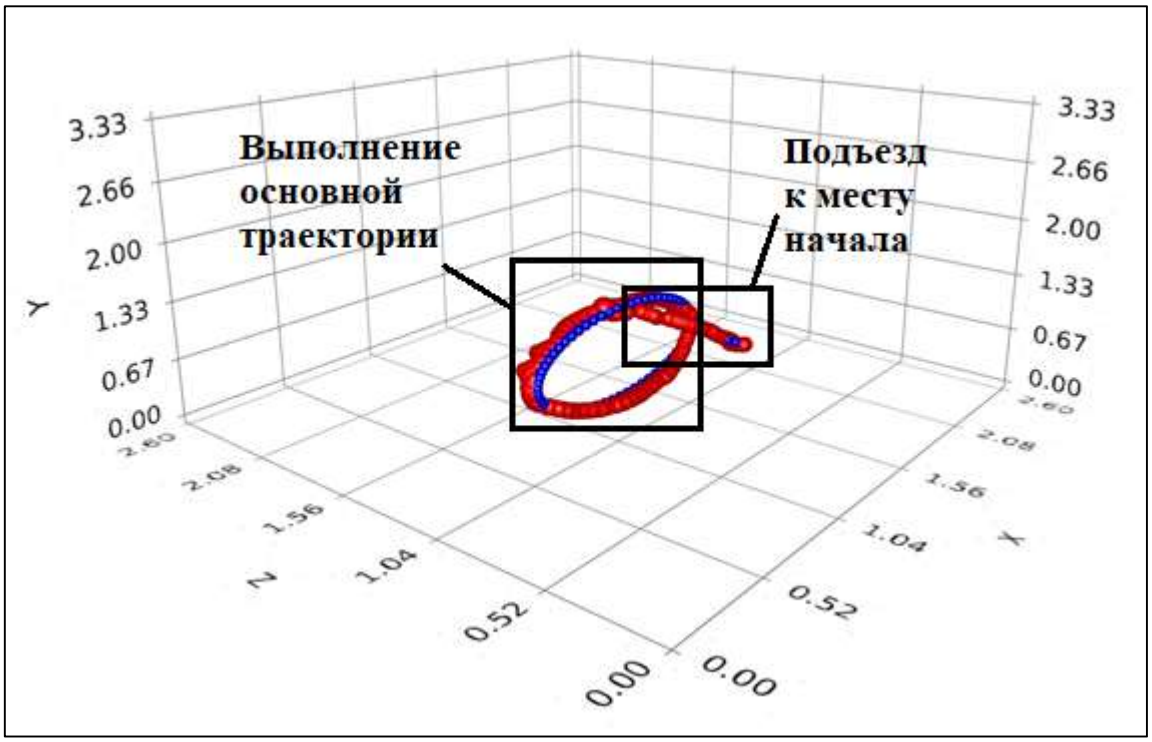


Рисунок 1.40 – Результат выполнения траектории

На рисунке 1.40 можно видеть два этапа выполнения траектории. Первым является подъезд к месту выполнения траектории. Программно этот этап является частью траектории и программа не замечает разницы между основной траекторией и подъездом. Вторым этапом является непосредственное выполнение траектории.

По заданной и действительной траектории было выполнено сравнение. Полученные данные в различных проекциях представлены ниже (рисунки 1.41, 1.42, 1.43).

Как можно заметить на проекции XY траектории уставки и оценки практически совпадают, максимальное отклонение составляет порядка 3-4 см. Однако, на проекциях XZ и YZ заметно значительное отклонение объекта по высоте. Это может говорить о необходимости проверки системы по поводу прямого и обратного трилатерационного преобразования. В частности, необходим дополнительный анализ трилатерационных преобразований. Закономерность особенно заметна на проекции YZ, где при удалении от центра происходит занижение.

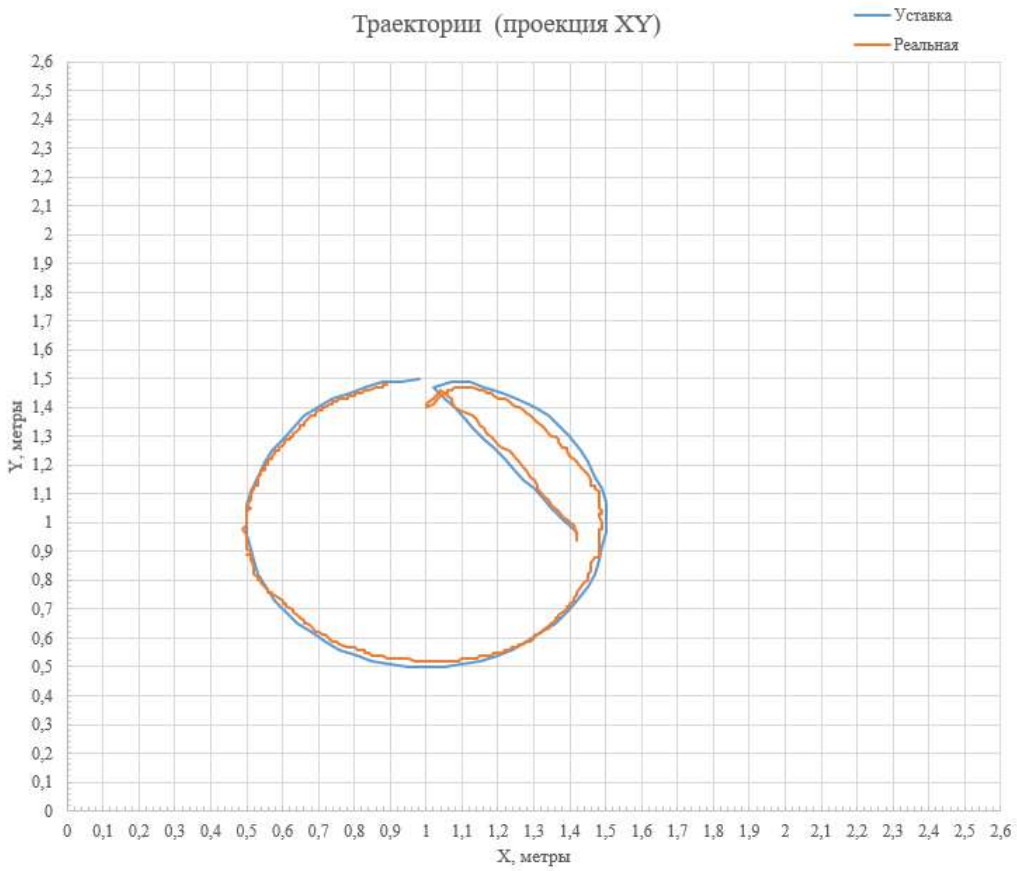


Рисунок 1.41 – Сравнение уставки и оцененной траектории в проекции XY

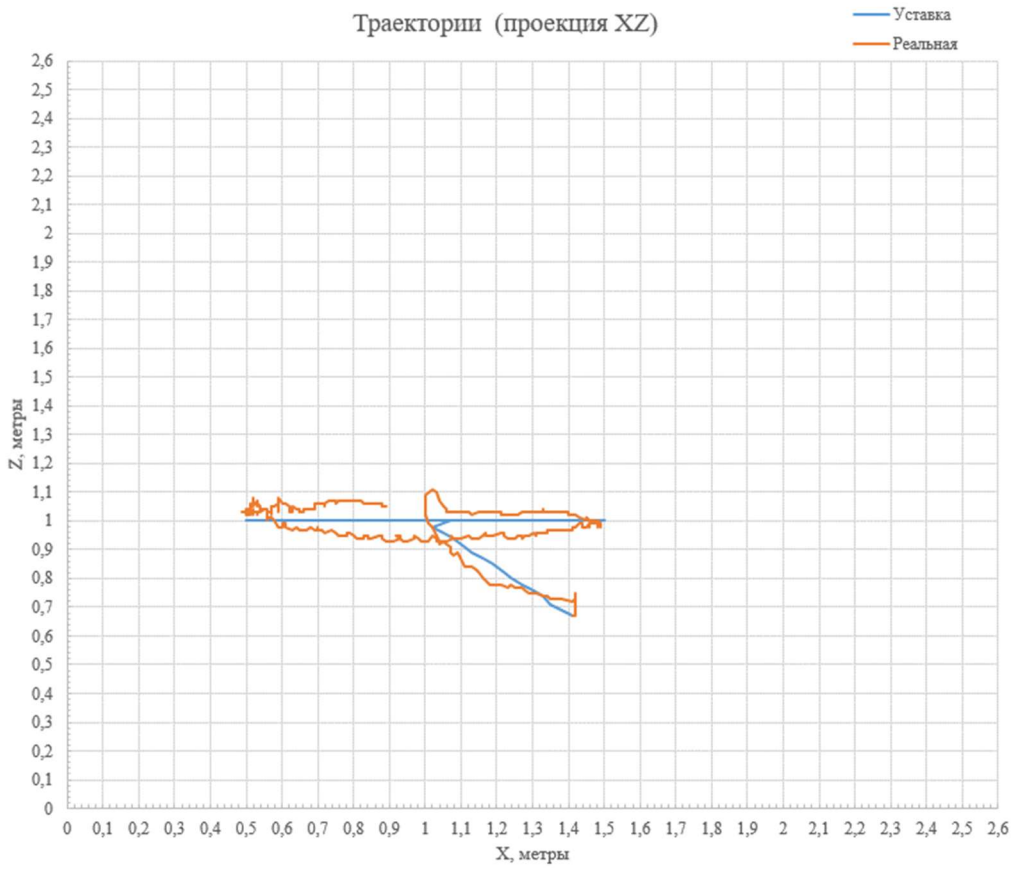


Рисунок 1.42 – Сравнение уставки и оцененной траектории в проекции XZ

Рисунок 1.43 – Сравнение уставки и оцененной траектории в проекции YZ

Поскольку основное программное обеспечение также предоставляет возможность получения данных с двигателей, был проведен анализ по собранным данным. Графики изменения углового положения выходных валов двигателей представлен ниже (рисунки 1.44-1.47).

На графиках цифрой 1 отмечен промежуток подъезда к точке начала выполнения траектории. Цифрой 2 отмечено выполнение непосредственно самой траектории.

Рисунок 1.44 – Положение вала двигателя 1 от времени

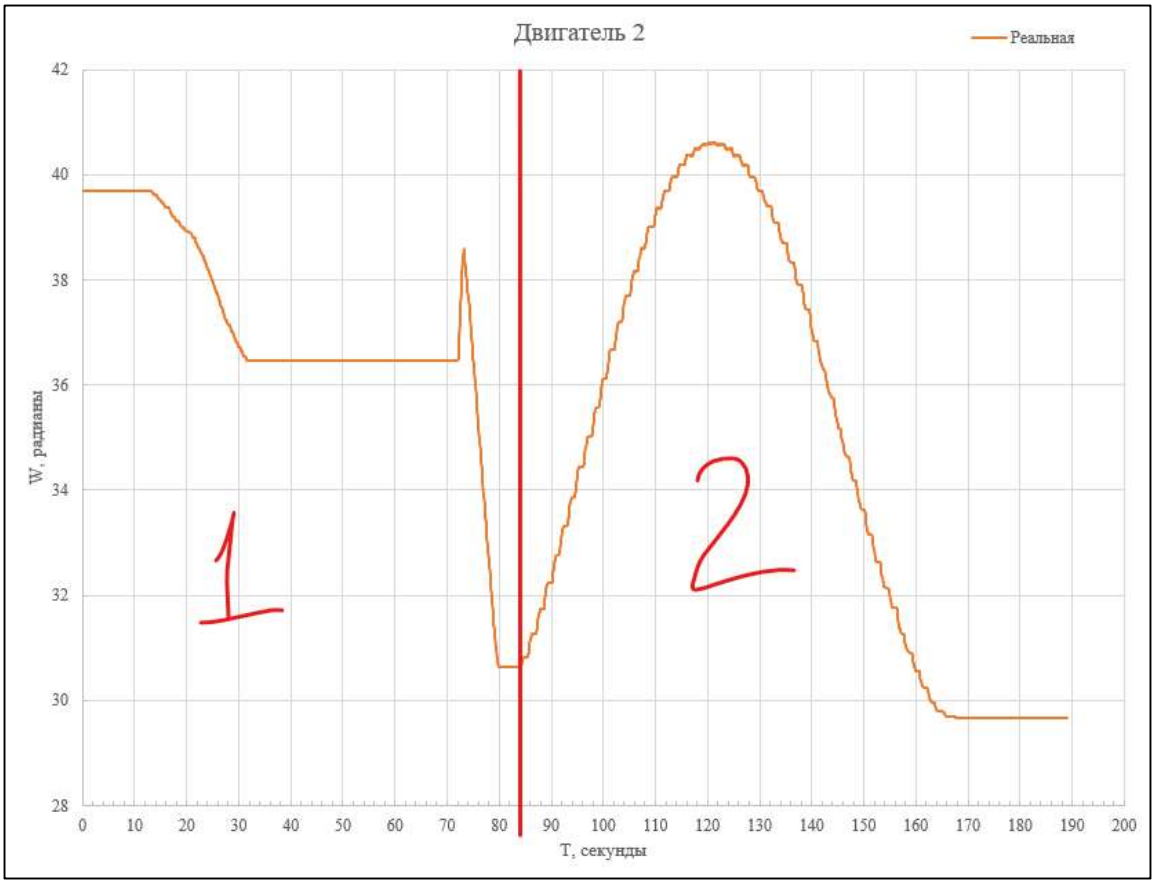


Рисунок 1.45 – Положение вала двигателя 2 от времени

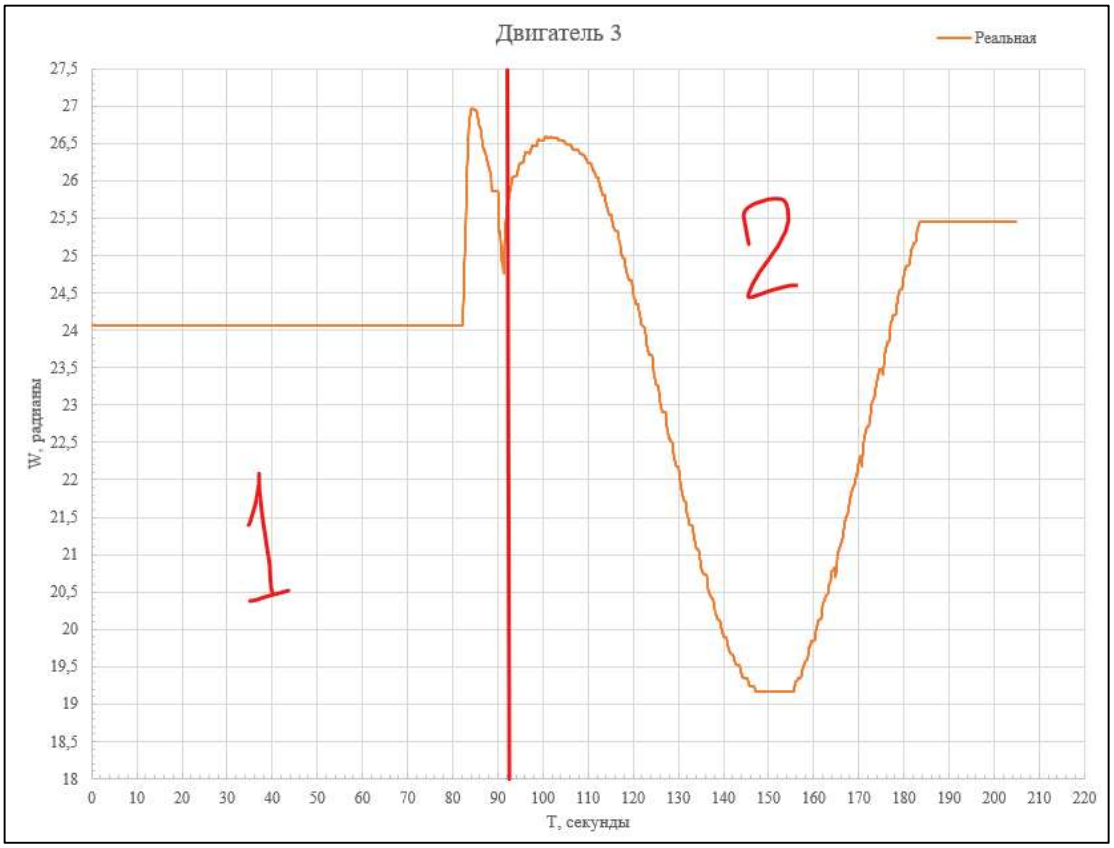


Рисунок 1.46 – Положение вала двигателя 3 от времени

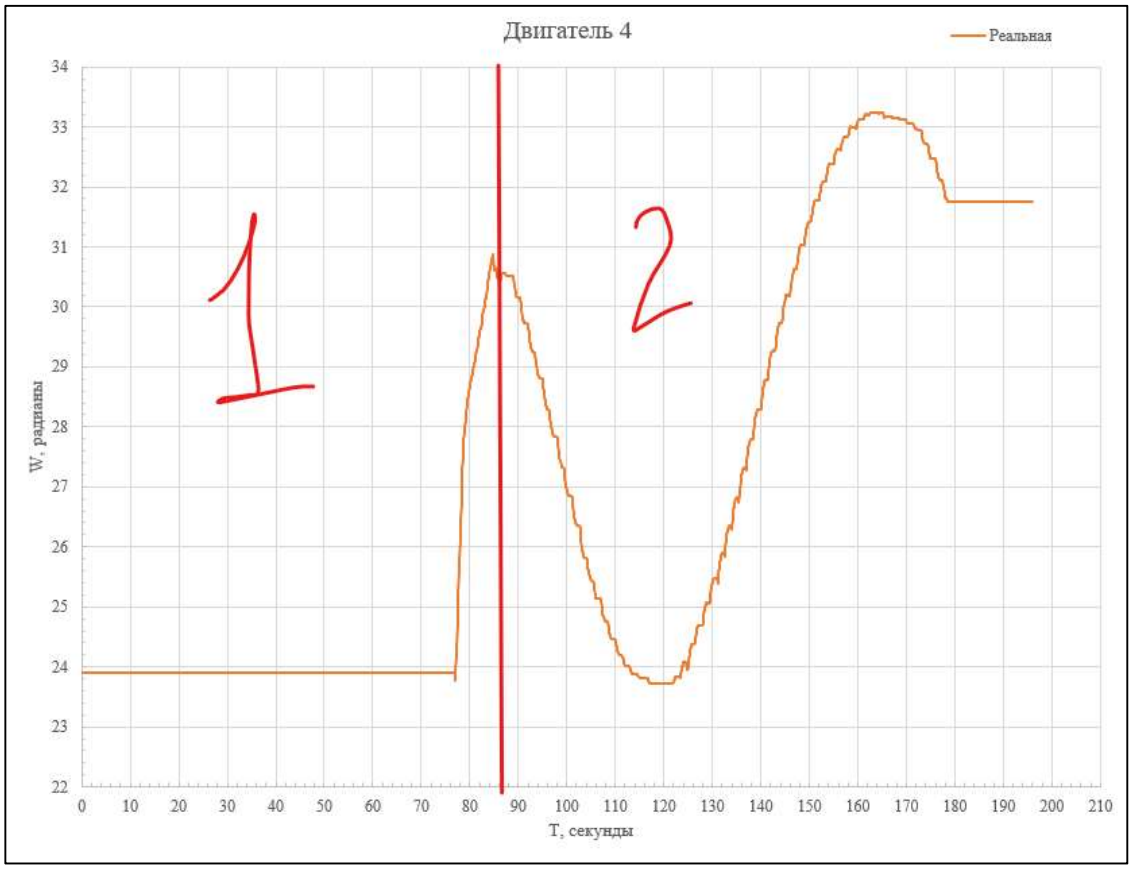


Рисунок 1.47 – Положение вала двигателя 4 от времени

По выше приведенным графикам можно действительно говорить о том, что характер траектории должен выполняться верно. Все графики на этапе выполнения траектории описывают четкий период синусоиды, что говорит о том, что двигатель совершает выпуск троса и его возвращение в изначальную длину.

Однако, на графиках возможно заметить шипообразные всплески во время выполнения траектории. Примеры подобных всплесков продемонстрированы на рисунке 1.48. Рисунок 1.48 представлен ниже.

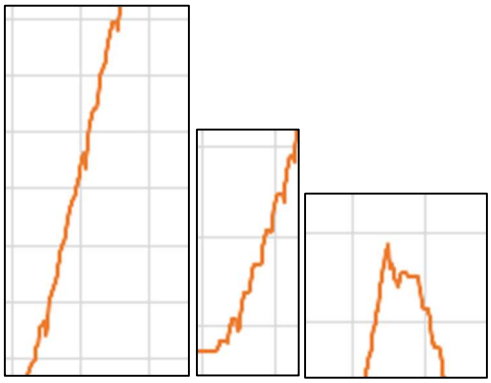


Рисунок 1.48 – Шипообразные всплески при выполнении траектории

Данные всплески не могут быть обусловлены системой управления, так как перерегулирование и колебательность у системы значительно ниже. На текущий момент данные всплески связываются с влиянием импульсных помех на канал энкодера. Данная проблема в значительной степени может влиять на всю систему и вносить помехи. В том числе, это влияет на систему оценки положения.

Решением данной проблемы может быть введение дифференциального приемо-передатчика на каждом канале энкодера.

ЗАДАНИЕ К РАЗДЕЛУ
«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Обучающемуся:

Группа	ФИО
8ЕМ11	Иванову Егору Андреевичу

Школа	ИШИТР	Отделение (НОЦ)	Отделение автоматизации и робототехники
Уровень образования	Магистратура	Направление/ специальность	15.04.06 Мехатроника и робототехника

Исходные данные к разделу «Социальная ответственность»:	
1. Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика) и области его применения.	Объект исследования: система 4-тросового обезвешивания. Область применения: робототехника. Рабочее место: лаборатория. Количество и наименование оборудования рабочей зоны: персональный компьютер, система тросового обезвешивания, испытательный полигон.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Производственная безопасность 1.1. Анализ выявленных вредныхфакторов: – Природа воздействия; – Действие на органы человека; – Нормы воздействия и нормативные документы (для вредных факторов); – СИЗ коллективные и индивидуальные; 1.2. Анализ выявленных опасных факторов: – Термических источники опасности; – Электробезопасность; – Пожаробезопасность.	1. Вредные факторы: 1.1. Недостаточная освещённость; 1.2. Нарушение микроклимата, оптимальные и допустимые параметры; 1.3. Шум, ПДУ, СКЗ, СИЗ; 1.4. Повышенный уровень электромагнитного излучения, ПДУ, СКЗ, СИЗ; 2. Опасные факторы: 2.1. Электроопасность, класс электроопасности помещения, безопасные номиналы I, U, $R_{заземления}$, СКЗ, СИЗ. Приведён расчёт освещения рабочего места; 2.2. Пожароопасность, категория пожароопасности помещения, марки огнетушителей, их назначение и ограничение применения. Приведена схема эвакуации.
2. Экологическая безопасность: – Выбросы в окружающую среду; – Решения по обеспечению экологической безопасности.	Наличие промышленных отходов (бумага-черновики, пластмасса, перегоревшие люминесцентные лампы, оргтехника, свинцовые и литий-ионные аккумуляторы) и способы их утилизации.
3. Безопасность в чрезвычайных ситуациях: 1. Перечень возможных ЧС при разработке и эксплуатации проектируемого решения; 2. Разработка превентивных мер по предупреждению ЧС; 3. Разработка действий в результате возникшей ЧС и мер по ликвидации её последствий.	Рассмотрены 2 ситуации ЧС: 1. Природная – сильные морозы зимой (аварии на электро-, тепло- коммуникациях, водоканале, транспорте); 2. Техногенная – несанкционированное проникновение посторонних на рабочее место (возможны проявления вандализма, диверсии, промышленного шпионажа), представлены мероприятия по обеспечению устойчивой работы производства в том и другом случае.
4. Перечень нормативно-технической документации	ГОСТы, СанПиНы, СНиПы.
Дата выдачи задания для раздела по линейному графику	06.05.2023

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор ООД ШБИП	Федорчук Ю.М.	д-р техн. наук, профессор		06.05.2023

Задание принял к исполнению обучающийся:

Группа	ФИО	Подпись	Дата
8ЕМ11	Иванов Егор Андреевич		06.05.2023

2 Социальная ответственность

Социальная ответственность - ответственность отдельного ученого и научного сообщества перед обществом. Первостепенное значение при этом имеет безопасность применения технологий, которые создаются на основе достижений науки, предотвращения или минимизации возможных негативных последствий их применения, обеспечение безопасного как для испытуемых, как и для окружающей среды проведения исследований.

В рамках данной работы было проведено исследование управления 4-х тросовой системой тросового обезвешивания. Исследование проводилось в лаборатории в аудитории 101а 10-го корпуса НИ ТПУ. Раздел также включает в себя оценку условий труда на рабочем месте, анализ вредных и опасных факторов труда, разработку мер защиты от них.

2.1 Вредные факторы

2.1.1 Отклонения показателей микроклимата в помещении

Проанализируем микроклимат в помещении, где находится рабочее место. Микроклимат производственных помещений определяют следующие параметры: температура, относительная влажность, скорость движения воздуха. Эти факторы влияют на организм человека, определяя его самочувствие.

Оптимальные и допустимые значения параметров микроклимата приведены в таблице 2.1 и 2.2.

Таблица 2.1 – Оптимальные нормы микроклимата

Период года	Температура воздуха, °С	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	19-23	40-60	0,1
Тёплый	23-25		0,2

Таблица 2.2 – Допустимые нормы микроклимата

Период года	Температура воздуха, °С		Относительная влажность воздуха, %	Скорость движения воздуха, м/с
	верхняя граница	нижняя граница		
Холодный	15	24	20-80	< 0,5
Тёплый	22	28	20-80	< 0,5

Общая площадь рабочего помещения составляет 42м^2, объем составляет 147м^3. По СанПиН 2.2.2/2.4.1340-03 санитарные нормы составляют 6,5 м^2 и 20 м^{3}объема на одного человека. Исходя из приведенных выше данных, можно сказать, что количество рабочих мест соответствует размерам помещения по санитарным нормам.

После анализа габаритных размеров рассмотрим микроклимат в этой комнате. В качестве параметров микроклимата рассмотрим температуру, влажность воздуха, скорость ветра.

В помещении осуществляется естественная вентиляция посредством наличия легко открываемого оконного проема (форточки), а также дверного проема. По зоне действия такая вентиляция является общеобменной. Основной недостаток - приточный воздух поступает в помещение без предварительной очистки и нагревания. Согласно норм СанПиН 2.2.2/2.4.1340-03 объем воздуха необходимый на одного человека в помещении без дополнительной вентиляции должен быть более 40м^3 [10]. В нашем случае объем воздуха на одного человека составляет 42 м^3, из этого следует, что дополнительная вентиляция не требуется. Параметры микроклимата поддерживаются в холодное время года за счет систем водяного отопления с нагревом воды до 100°С, а в теплое время года – за счет кондиционирования, с параметрами согласно [11]. Нормируемые параметры микроклимата, ионного состава воздуха, содержания вредных веществ должны соответствовать требованиям [12].

2.1.2 Превышение уровней шума

Одним из наиболее распространенных в производстве вредных факторов является шум. Он создается вентиляционным и рабочим оборудованием, преобразователями напряжения, рабочими лампами дневного света, а также проникает снаружи. Шум вызывает головную боль, усталость, бессонницу или сонливость, ослабляет внимание, память ухудшается, реакция уменьшается.

Основным источником шума в комнате являются компьютерные охлаждающие вентиляторы и. Уровень шума варьируется от 35 до 42 дБА. Согласно СанПиН 2.2.2 / 2.4.1340-03, при выполнении основных работ на ПЭВМ уровень шума на рабочем месте не должен превышать 82 дБА [10].

При значениях выше допустимого уровня необходимо предусмотреть средства индивидуальной защиты(СИЗ) и средства коллективной защиты (СКЗ) от шума.

Средства коллективной защиты:

1) устранение причин шума или существенное его ослабление в источнике образования;
2) изоляция источников шума от окружающей среды (применение глушителей, экранов, звукопоглощающих строительных материалов, например, любой пористый материал – шамотный кирпич, микропористая резина, поролон и др.);
3) применение средств, снижающих шум и вибрацию на пути их распространения;

Средства индивидуальной защиты: применение спецодежды и защитных средств органов слуха: наушники, беруши, антифоны.

2.1.3 Повышенный уровень электромагнитных излучений

Источником электромагнитных излучений в нашем случае являются дисплеи ПЭВМ. Монитор компьютера включает в себя излучения рентгеновской, ультрафиолетовой и инфракрасной области, а также широкий диапазон электромагнитных волн других частот. Согласно СанПиН 2.2.2/2.4.1340-03 напряженность электромагнитного поля по электрической составляющей на расстоянии 50 см вокруг ВДТ не должна превышать 25В/м в диапазоне от 5Гц до 2кГц, 2,5В/м в диапазоне от 2 до 400кГц [13]. Плотность магнитного потока не должна превышать в диапазоне от 5 Гц до 2 кГц 250нТл, и 25нТл в диапазоне от 2 до 400кГц. Поверхностный электростатический потенциал не должен превышать 500В [13]. В ходе работы использовалась ПЭВМ типа ноутбук со следующими характеристиками: напряженность

электромагнитного поля 2,5В/м; поверхностный потенциал составляет 450 В (основы противопожарной защиты предприятий ГОСТ 12.1.004 и ГОСТ 12.1.010 – 76).

При длительном постоянном воздействии электромагнитного поля (ЭМП) радиочастотного диапазона при работе на ПЭВМ у человеческого организма сердечно-сосудистые, респираторные и нервные расстройства, головные боли, усталость, ухудшение состояния здоровья, гипотония, изменения сердечной мышцы проводимости. Тепловой эффект ЭМП характеризуется увеличением температуры тела, локальным селективным нагревом тканей, органов, клеток за счет перехода ЭМП на теплую энергию.

Предельно допустимые уровни (ПДУ) облучения (по *ГОСТ 54 30013-83)*:

1) до 10 мкВт/см2, время работы (8 часов);

2) от 10 до 100 мкВт/см2, время работы не более 2 часов;

3) от 100 до 1000 мкВт/см2, время работы не более 20 мин. при условии пользования защитными очками;

4) для населения в целом ППМ не должен превышать 1 мкВт/см2.

Защита человека от опасного воздействия электромагнитного излучения осуществляется следующими способами:

СКЗ:

1) защита временем;
2) защита расстоянием;
3) снижение интенсивности излучения непосредственно в самом источнике излучения;
4) заземление экрана вокруг источника;
5) защита рабочего места от излучения;

СИЗ: очки и специальная одежда, выполненная из металлизированной ткани (кольчуга). При этом следует отметить, что использование СИЗ возможно при кратковременных работах и является мерой аварийного

характера. Ежедневная защита обслуживающего персонала должна обеспечиваться другими средствами.

Вместо обычных стекол используют стекла, покрытые тонким слоем золота или диоксида олова (SnO2).

2.1.4 Недостаточная освещённость

Для обеспечения требуемой освещенности необходимо использовать совмещенное освещение, создаваемое сочетанием естественного и искусственного освещения. При данном этапе развития осветительной техники целесообразно использовать люминесцентные лампы, которые посравнению с лампами накаливания имеют большую светоотдачу на ватт потребляемой мощности и более естественный спектр.

Минимальный уровень средней освещенности на рабочих местах с постоянным пребыванием людей должен быть не менее 200 лк.

В расчётном задании должны быть решены следующие вопросы:

1) выбор системы освещения;
2) выбор источников света;
3) выбор светильников и их размещение;
4) выбор нормируемой освещённости;
5) расчёт освещения методом светового потока.

В данном расчётном задании для всех помещений рассчитывается общее равномерное освещение.

Таблица 2.3 – Параметры помещения и выбранных ламп

Параметр	Обозначение	Значение
Длина помещения	A	8 м
Ширина помещения	B	8,4 м
Высота помещения	H	3,3 м
Свес	h_c	0 м
Высота рабочей поверхности	h_{pn}	0,75 м
Высота от светильников до рабочей поверхности	h	2,55 м
Коэффициент отражения стен	ρ_{cm}	70 %
Коэффициент отражения потолка	ρ_n	70 %
Коэффициент запаса	$k_з$	1,5
Коэффициент неравномерности	Z	1,1

Свес ламп был принят равным 0, так как лампы в лаборатории встраиваются в панельный потолок.

Выберем светильник типа ШОД, так как они подходят по наименьшей допустимой высоте подвеса над полом (2,5 м). Тогда расстояние между рядами

$$L = \lambda * h = 1,2 * 2,55 = 3,06 \text{ м},$$

а расстояние от крайних светильников или рядов до стены равно

$$l = L / 3 = 3,57 / 3 = 1,02 \text{ м}$$

Размещаем светильники в три ряда. Расположим ряды немного плотнее, так как ширины помещения не хватает

$$L = \frac{B - 2l}{3 - 1} = \frac{8 - 2 * 1,02}{2} = 2,98 \text{ м}$$

В ряду можно установить 4 светильника типа ШОД мощностью 40 Вт и длиной 1,228 м. При этом разрывы между светильниками в ряду составят 48,3 см.

Учитывая, что в каждом светильнике установлено две лампы, общее количество ламп для освещения лаборатории будет равно $N = 24$.

Находим индекс помещения

$$i = \frac{S}{h(A + B)} = \frac{67,2}{2,55 * (8 + 8,4)} = 1,61.$$

Коэффициент использования светового потока для светильников типа ШОД при индексе помещения 1,61 будет равен 0,51. Тогда расчётный световой поток лампы будет равен

$$\Phi = \frac{E_н * S * k_з * Z}{N * \eta} = \frac{300 * 67,2 * 1,5 * 1,1}{24 * 0,51} = 2717,6 \text{ лм}.$$

Ближайшей по световому потоку люминесцентной лампой мощностью 40 Вт является лампа ЛБ-40 со световым потоком 2800 лм.

Проверяем выполнение условия

$$-10\% \le \frac{\Phi_{л.станд} - \Phi_{л.расч}}{\Phi_{л.станд}} * 100\% \le +20\%,$$

$$-10\% \le 3,03\% \le +20\%,$$

В заключении, определим общую электрическую мощность получившейся осветительной установки

$$P = N * P_л = 24 * 40 = 960 \text{ Вт.}$$

По итогу расчёта получили следующую схему размещения светильников в лаборатории, которая приведена на рисунке 2.1.

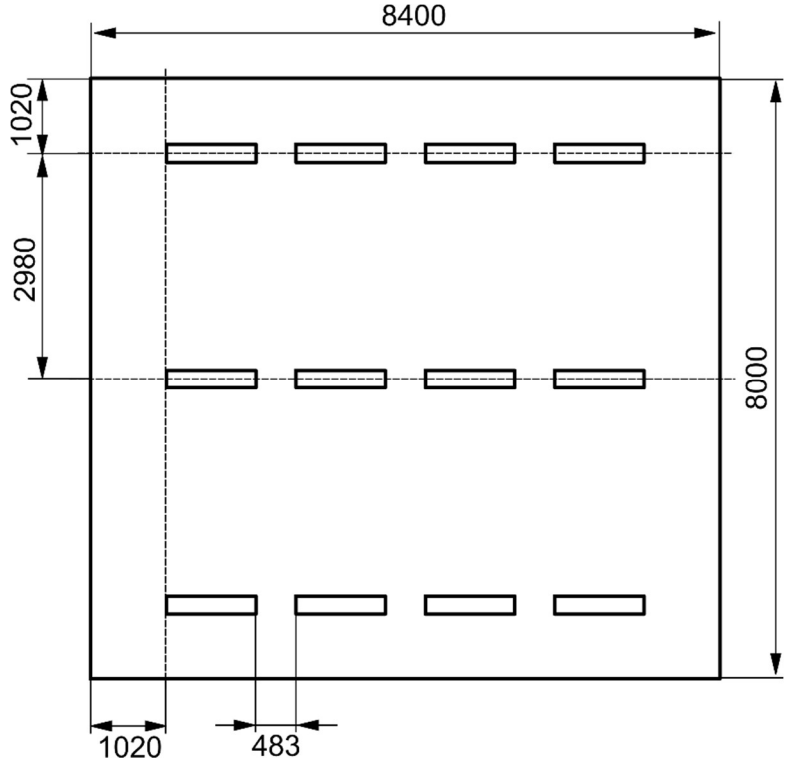


Рисунок 2.1 – План размещения светильников в лаборатории

2.2 Опасные факторы

2.2.1 Электроопасность, класс электроопасности помещения, безопасные номиналы I, U, $R_{заземления}$, СКЗ, СИЗ

К опасным факторам можно отнести наличие в помещении большого количества аппаратуры, использующей однофазный электрический ток напряжением 220 В и частотой 50Гц. По опасности электропоражения комната относится к помещениям без повышенной опасности, так как отсутствует повышенная влажность, высокая температура, токопроводящая пыль и возможность одновременного сприкосновения токоведущих элементов с заземленными металлическими корпусами оборудования [16].

Лаборатория относится к помещению без повышенной опасности поражения электрическим током. Безопасными номиналами являются: I < 0,1 А; U < (2-36) В; $R_{зазем}$ < 4 Ом.

Для защиты от поражения электрическим током используют СИЗ и СКЗ.

Средства коллективной защиты:

1) защитное заземление, зануление [17];
2) малое напряжение;
3) электрическое разделение сетей;
4) защитное отключение;
5) изоляция токоведущих частей;
6) оградительные устройства.
7) Использование щитов, барьеров, клеток, ширм, а также заземляющих и шунтирующих штанг, специальных знаков и плакатов.

Средства индивидуальной защиты: использование диэлектрических перчаток, изолирующих клещей и штанг, слесарных инструментов с изолированными рукоятками, указатели величины напряжения, калоши, боты, подставки и коврики.

2.2.2 Пожароопасность, категория пожароопасности помещения, марки огнетушителей, их назначение и ограничение применения;

Приведена схема эвакуации.

По взрывопожарной и пожарной опасности помещения подразделяются на категории А, Б, В1-В4, Г и Д.

Согласно НПБ 105-03 лаборатория относится к категории В– горючие и трудно горючие жидкости, твердые горючие и трудно горючие вещества и материалы, вещества и материалы, способные при взаимодействии с водой, кислородом воздуха или друг с другом только гореть, при условии, что помещения, в которых находится, не относятся к категории наиболее опасных А или Б.

По степени огнестойкости данное помещение относится к 1-й степени огнестойкости по СНиП 2.01.02-85 (выполнено из кирпича, которое относится к трудносгораемым материалам).

Возникновение пожара при работе с электронной аппаратурой может быть по причинам как электрического, так и неэлектрического характера.

Причины возникновения пожара неэлектрического характера: халатное неосторожное обращение с огнем (курение, оставленные без присмотра нагревательные приборы, использование открытого огня);

Причины возникновения пожара электрического характера: короткое замыкание, перегрузки по току, искрение и электрические дуги, статическое электричество и т. п.

Для локализации или ликвидации загорания на начальной стадии используются первичные средства пожаротушения. Первичные средства пожаротушения обычно применяют до прибытия пожарной команды.

Огнетушители водо-пенные (ОХВП-10) используют для тушения очагов пожара без наличия электроэнергии. Углекислотные (ОУ-2) и порошковые огнетушители предназначены для тушения электроустановок, находящихся под напряжением до 1000В. Для тушения токоведущих частей и электроустановок применяется переносной порошковый огнетушитель, например ОП-5.

В общественных зданиях и сооружениях на каждом этаже должно размещаться не менее двух переносных огнетушителей. Огнетушители следует располагать на видных местах вблизи от выходов из помещений на высоте не более 1,35 м. Размещение первичных средств пожаротушения в коридорах, переходах не должно препятствовать безопасной эвакуации людей.

Для предупреждения пожара и взрыва необходимо предусмотреть:

1) специальные изолированные помещения для хранения и разлива легковоспламеняющихся жидкостей (ЛВЖ), оборудованные приточно-вытяжной вентиляцией во взрывобезопасном исполнении - соответствии с ГОСТ 12.4.021-75 и СНиП 2.04.05-86;

2) специальные помещения (для хранения в таре пылеобразной канифоли), изолированные от нагревательных приборов и нагретых частей оборудования;
3) первичные средства пожаротушения на производственных участках (передвижные углекислые огнетушители ГОСТ 9230-77, пенные огнетушители ТУ 22-4720-80, ящики с песком, войлок, кошма или асбестовое полотно);
4) автоматические сигнализаторы (типа СВК-3 М 1) для сигнализации о присутствии в воздухе помещений предвзрывных концентраций горючих паров растворителей и их смесей.

Лаборатория полностью соответствует требованиям пожарной безопасности, а именно, наличие охранно-пожарной сигнализации, плана эвакуации, изображенного на рисунке 2.2, порошковых огнетушителей с поверенным клеймом, табличек с указанием направления к запасному (эвакуационному) выходу.

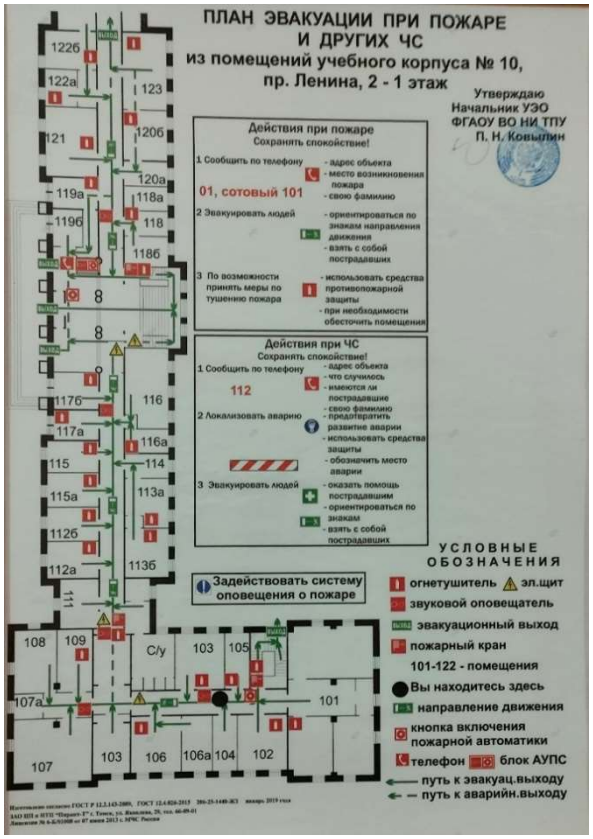


Рисунок 1.2 – План эвакуации при пожаре и других ЧС из помещений учебного корпуса №10 НИ ТПУ

2.3 Экологическая безопасность

В компьютерах огромное количество компонентов, которые содержат токсичные вещества и представляют угрозу, как для человека, так и для окружающей среды.

К таким веществам относятся:

1) свинец (накапливается в организме, поражая почки, нервную систему);

2) ртуть(поражает мозг и нервную систему);

3) никель и цинк (могут вызывать дерматит);

4) щелочи (прожигают слизистые оболочки и кожу);

Поэтому компьютер требует специальных комплексных методов утилизации.

Таким образом утилизацию компьютера можно провести следующим образом:

1) отделить металлические детали от неметаллов;

2) разделить углеродистые металлы от цветмета;

3) пластмассовые изделия (крупногабаритные) измельчить для уменьшения объема;

4) копир-порошок упаковать в отдельную упаковку, точно также, как и все проклассифицированные и измельченные компоненты оргтехники, и после накопления на складе транспортных количеств отправить предприятиям и фирмам, специализирующимся по переработке отдельных видов материалов.

Люминесцентные лампы утилизируют следующим образом. Не работающие лампы немедленно после удаления из светильника должны быть упакованы в картонную коробку, бумагу или тонкий мягкий картон, предохраняющий лампы от взаимного соприкосновения и случайного механического повреждения. После накопления ламп объемом в 1 транспортную единицу их сдают на переработку на соответствующее предприятие. Недопустимо выбрасывать отработанные энергосберегающие лампы вместе с обычным мусором, превращая его в ртутьсодержащие отходы, которые загрязняют ртутными парами.

2.4 Безопасность в чрезвычайных ситуациях

Природная чрезвычайная ситуация – обстановка на определенной территории или акватории, сложившейся в результате возникновения источника природной чрезвычайной ситуации, который может повлечь или повлек за собой человеческие жертвы, ущерб здоровью людей и (или) окружающей природной среде, значительные материальные потери и нарушение условий жизнедеятельности людей.

Лаборатория расположена в городе Томске с континентально-циклоническим климатом. Такие природные явления, как землетрясения, наводнения, засухи, ураганы и т. д., в данном регионе маловероятны.

Возможными ЧС на объекте в данном случае, могут быть сильные морозы. Для Сибири в зимнее время года характерны сильные морозы. Достижение критически низких температур приводит к авариям систем тепло- и водоснабжения, сантехнических коммуникаций и электроснабжения, приостановке работы. В этом случае при подготовке к зиме следует предусмотреть

1) газобаллонные калориферы (запасные обогреватели),
2) дизель или бензоэлектрогенераторы,
3) запасы питьевой и технической воды на складе,
4) теплый транспорт для доставки работников на работу и с работы домой в случае отказа муниципального транспорта.

Их количества и мощности должно хватать для того, чтобы работа на производстве не прекратилась.

В лаборатории outdoor мобильной робототехники наиболее вероятно возникновение чрезвычайных ситуаций (ЧС) техногенного характера. Для предупреждения вероятности осуществления диверсии предприятие оборудовано системой видеонаблюдения, круглосуточной охраной, пропускной системой, надежной системой связи. Должностные лица раз в полгода проводят тренировки по отработке действий на случай экстренной эвакуации.

ЗАДАНИЕ К РАЗДЕЛУ
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Обучающемуся:

Группа	ФИО
8ЕМ11	Иванову Егору Андреевичу

Школа	ИШИТР	Отделение школы (НОЦ)	ОАР
Уровень образования	Магистратура	Направление/специальность	15.04.06 Мехатроника и робототехника

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. *Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих*	Стоимость ресурсов определялась по средней рыночной стоимости, в соответствии с окладами сотрудников организации.
2. *Нормы и нормативы расходования ресурсов*	Тариф на электроэнернию 3,16 кВт/ч. 30% районный коэффициент
3. *Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования*	Отчисления в социальные внебюджетные фонды 30 %.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. *Оценка коммерческого и инновационного потенциала НТИ*	Провести предпроектный анализ
2. *Разработка устава научно-технического проекта*	Представить Устав научного проекта магистерской работы
3. *Планирование процесса управления НТИ: структура и график проведения, бюджет, риски и организация закупок*	Разработать план управления НТИ
4. *Определение ресурсной, финансовой, экономической эффективности*	Рассчитать сравнительную эффективность исследования
Перечень графического материала *(с точным указанием обязательных чертежей):*	
1. Оценочная карта для сравнения конкурентных технических решений. 2. Интерактивная матрица сильных сторон и возможностей проекта. 3. Интерактивная матрица сильных сторон и угроз проекта. 4. Интерактивная матрица слабых сторон и возможностей проекта. 5. Интерактивная матрица слабых сторон и угроз проекта. 6. SWOT-анализ. 7. Оценка готовности проекта к коммерциализации. 8. Заинтересованные стороны проекта. 9.Цели и результаты проекта. 10. Рабочая группа проекта. 11. Календарный план проекта. 12. Календарный план-график проведения НИОКР по теме. 13. Группировка затрат по статьям. 14. Сырьё, материалы, комплектующие изделия и покупные полуфабрикаты. 15. Расчёт затрат по статье спецоборудование для научных работ для первого варианта исполнения проекта. 16. Расчёт затрат по статье спецоборудование для научных работ для второго варианта исполнения проекта. 17. Расчёт основной заработной платы и отчисления на социальные нужды для первого варианта исполнения. 18. Расчёт основной заработной платы и отчисления на социальные нужды для второго варианта исполнения. 19. Сравнительная оценка характеристик вариантов исполнения проекта. 20. Сравнительна эффективность разработки	

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОСГН ШБИП	Былкова Татьяна Васильевна	канд. экон. наук, доцент		

Задание принял к исполнению обучающийся:

Группа	ФИО	Подпись	Дата
8ЕМ11	Иванов Егор Андреевич		

3 Финансовый менеджмент

3.1 Предпроектный анализ

3.1.1 Анализ конкурентных технических решений с позиции ресурсоэффективности и ресурсосбережения

В качестве конкурентного решения было взято устройство тросового обезвешивания под названием FLOAT.

Оценочная карта для сравнения конкурентных технических решений представлена в таблице 3.1. Таблица 3.1 представлена ниже.

Таблица 3.1 – Оценочная карта для сравнения конкурентных технических решений

Критерии оценки	**Вес критерия**	**Баллы**		**Конкурентоспособность**	
		$Б_ф$	$Б_{к1}$	$К_ф$	$К_{к1}$
Технические критерии оценки ресурсоэффективности					
1. Повышение труда и производительности	0,05	4	4	0,20	0,204,
2. Удобство в эксплуатации (соответствует требованиям потребителей)	0,15	5	4	0,75	0,60
3. Энергоэкономичность	0,05	4	3	0,20	0,15
4. Надежность	0,15	4	5	0,60	0,75
5. Уровень шума	0,05	3	4	0,15	0,20
6. Безопасность	0,15	4	4	0,60	0,60
7. Функциональная мощность (предоставляемые возможности)	0,10	4	4	0,40	0,40
8. Простота эксплуатации	0,10	5	3	0,50	0,30
9. Качество интеллектуального интерфейса	0,10	4	4	0,40	0,40
10. Возможность подключения в сеть ЭВМ	0,10	5	5	0,50	0,50
Итого	**1**	42	40	4,30	4,10

В ходе анализа выяснилось, что конкурентоспособность научной разработки выше, чем у указанного конкурента. Это может быть обусловлено большей простотой эксплуатации, дружелюбностью пользовательского интефейса, что также сказывается на удобстве эксплуатации продукта конечным пользователем.

3.1.2 SWOT-анализ

Были сформированы интерактивные матрицы проекта. Интерактивные матрицы проекта представлены в таблицах 3.2-3.5. Таблицы 3.2-3.5 представлены ниже.

Таблица 3.2 – Интерактивная матрица сильных сторон и возможностей проекта

Сильные стороны проекта				
Возможности проекта		С1	С2	С3
	В1	+	-	-
	В2	+	+	+
	В3	-	+	-

Таблица 3.3 – Интерактивная матрица сильных сторон и угроз проекта

Сильные стороны проекта				
Угрозы проекта		С1	С2	С3
	У1	+	0	0
	У2	+	0	0
	У3	+	0	0

Таблица 3.4 – Интерактивная матрица слабых сторон и возможностей проекта

Слабые стороны проекта				
Возможности проекта		Сл1	Сл2	Сл3
	В1	0	+	0
	В2	+	-	+
	В3	-	+	+

Таблица 3.5 – Интерактивная матрица слабых сторон и угроз проекта

Слабые стороны проекта				
Угрозы проекта		Сл1	Сл2	Сл3
	У1	-	+	+
	У2	-	-	-
	У3	-	-	-

SWOT-анализ научно-исследовательского проекта представлен в таблице 3.6. Таблица 3.6 представлена ниже.

Таблица 3.6 – SWOT-анализ

	Сильные стороны научно-исследовательского проекта: С1. Низкая себестоимость С2. Простота эксплуатации С3. Низкое энергопотребление	**Слабые стороны научно-исследовательского проекта:** Сл1. Большой срок поставок материалов Сл2. Высокий уровень шума при работе Сл3. Низкая скорость работы
Возможности: В1. Использование инновационной инфраструктуры ТПУ В2. Появление дополнительного спроса на новый продукт В3. Повышение стоимости конкурентных разработок	С1В2, С2В2, С3В3 – Низкая себестоимость, простота эксплуатации и низкое энергопотребление могут привести к повышению спроса на продукт С2В3 – В погоне за большей простотой эксплуатации может повыситься стоимость конкурентных разработок С1В1 – Благодаря использованию инфраструктуры ТПУ может снизиться стоимость разработки, так как не придется затрачиваться на оборудование	Сл1В2 – Появление дополнительного спроса на продукт при задержке поставок может привести к негативным последствиям при взаимодействии с покупателем Сл2В1 – За счет использования оборудования ТПУ можно провести исследования по снижению шума Сл2В3 – За счет повышения стоимости конкурентов покупатели могут предпочесть более шумный, но менее дорогой аналог Сл3В2, Сл3В3 – За счет повышения стоимости конкурентов можно повысить стоимость и использовать компоненты для увеличения скорости работы системы, за счет повышения спроса можно вложить больше прибыли в исследования и повысить скорость работы
Угрозы: У1. Отсутствие спроса на новые технологии производства У2. Ограничения на экспорт технологии У3. Несвоевременное финансовое обеспечение научного исследования со стороны государства	С1У1, С1У2, С1У3 – За счет низкой себестоимости возможно снизить цену на продукт при выходе на рынок, что может привлечь клиентов. При ограничении на экспорт возможно повысить цену для продажи внутри страны, что принесет большую прибыль при низкой себестоимости. За счет низкой себестоимости можно без проблем пережить несвоевременное финансовое обеспечение	Сл2У1, Сл3У1 – При наличии высокого шума при работе и при низкой скорости работы спрос на продукт может быть низким

3.1.3 Оценка готовности проекта к коммерциализации

Оценка готовности проекта к коммерциализации представлена в таблице 3.7. Таблица 3.7 представлена ниже.

Таблица 3.7 – Оценка готовности проекта к коммерциализации

№	Наименование	Степень проработан-ности научного проекта	Уровень имеющихся знаний у разработ-чика
1.	Определен имеющийся научно-технический задел	5	5
2.	Определены перспективные направления коммерциализации научно-технического задела	5	4
3.	Определены отрасли и технологии для предложения	4	5
4.	Определена товарная форма научно-технического задела для представления на рынок	3	5
5.	Определены авторы и осуществлена охрана их прав	5	4
6.	Проведена оценка стоимости интеллектуальной собственности	4	5
7.	Проведены маркетинговые исследования рынков сбыта	5	4
8.	Разработан бизнес-план коммерциализации научной разработки	4	4
9.	Определены пути продвижения научной разработки на рынок	3	5
10.	Разработна стратегия реализации научной разработки	4	5
11.	Проработаны вопросы международного сотрудничества и выхода на зарубежный рынок	5	4
12.	Проработаны вопросы использования услуг инфраструктуры поддержки, получения льгот	5	5
13.	Проработаны вопросы финансирования коммерциализации научной разработки	4	5
14.	Имеется команда для коммерциализации научной разработки	5	4
15.	Проработан механизм реализации научного проекта	4	5
	ИТОГО БАЛЛОВ	65	69

Итоговая сумма баллов оказалась в промежутке от 60 до 75, следовательно, разработка является перспективной. Возможно улучшить определение

товарной формы научно-технического задела для представления на рынок, а также определение пути продвижения научной разработки на рынок.

3.1.4 Методы коммерциализации результатов научно-технического исследования

Из всех возможных методов коммерциализации результатов научно-технического исследования была выбрана торговля патентными лицензиями. Данный выбор обусловлен тем, что результатом научно-технического исследования является исследование методов обезвешивания и разработка программного обеспечения для обезвешивания. При этом данное ПО не является самостоятельным и обособленным, т.е. для его работы нужна установка по обезвешиванию. Разработка встраивается в уже готовую систему. Таким образом, патентная лицензия будет разрешать разработчикам и компаниям интегрировать авторское решение в свою систему.

3.2 Инициация проекта

Представим устав проекта:

1) В таблице 3.8 представлены заинтересованные стороны проекта. Таблица 3.8 представлена ниже.

Таблица 3.8 – Заинтересованные стороны проекта

Заинтересованные стороны проекта	Ожидания заинтересованных сторон
Роскосмос	Возможность проверки раскрытия солнечных панелей в наземных условиях
Министерство здравоохранения	Возможность лечения пациентов с опорно-двигательными проблемами

2) В таблице 3.9 представлены цели и результаты проекта. Таблица 9 представлена ниже.

Таблица 3.9 – Цели и результаты проекта

Цели проекта	Разработка и исследование программного обеспечения системы тросового обезвешивания
Ожидаемые результаты проекта	Программное обеспечение для обеспечения обезвешивания и перемещения объекта в пространстве

3) Рабочая группа проекта приведена в таблице 3.10. Таблица 3.10 представлена ниже.

Таблица 3.10 – Рабочая группа проекта

№ п/п	ФИО	Основное место работы, должность	Роль в проекте	Функции	Трудозатраты, час
1	Иванов Егор Андреевич	-	Исполнитель	Выполнение работы по проекту	768
2	Беляев Александр Сергеевич	НИ ТПУ, ОАР ИШИТР, старший преподаватель	Эксперт	Консультирует по конкретным вопросам	40
3	Филипас Александр Александрович	НИ ТПУ, ОАР ИШИТР, заведующий кафедрой – руководитель отделения на правах кафедры	Руководитель	Координирует команду	20
				Итого	828

3.3 Планирование управления научно-техническим проектом

3.3.1 План проекта

Составим план проекта. В таблице 3.11 представлен линейный график проекта.

Таблица 3.11 – Календарный план проекта

№ п/п	Название	Длительность, дни	Дата начала	Дата окончания	Состав участников
1	Проведение литературного обзора по теме исследования	16	07.02.2023	25.02.2023	Исполнитель
2	Календарное планирование работ по теме	2	27.02.2023	28.02.2023	Руководитель, эксперт, исполнитель
3	Подготовка аппаратной части установки	16	01.03.2023	18.03.2023	Эксперт, исполнитель
4	Разработка архитектуры программного обеспечения	3	20.03.2023	22.03.2023	Исполнитель
5	Разработка загрузчика микроконтроллера	10	23.03.2023	01.04.2023	Эксперт, исполнитель
6	Разработка основного программного обеспечения микроконтроллера	12	03.04.2023	15.04.2023	Исполнитель
7	Разработка протоколов взаимодействия	12	17.04.2023	29.04.2023	Исполнитель
8	Разработка программного обеспечения ПК	16	02.05.2023	20.05.2023	Исполнитель
9	Проведение экспериментов с установкой обезвешивания	6	22.05.2023	27.05.2023	Исполнитель
10	Оценка полученных результатов	3	29.05.2023	31.05.2023	Руководитель, эксперт, исполнитель
Итог		96			

В соответствии с календарным планом проекта была составлена диаграмма Ганта. В таблице 3.12 показана диаграмма Ганта.

Таблица 3.12 – Календарный план-график проведения НИОКР по теме

№ работ	Вид работ	Исполнители	Тк, кал. дн.	Продолжительность выполнения работ
				февраль: 1, 2, 3; март: 1, 2, 3; апрель: 1, 2, 3; май: 1, 2, 3
1	Проведение литературного обзора по теме исследования	Исполнитель	16	
2	Календарное планирование работ по теме	Руководитель Эксперт Исполнитель	2	
3	Подготовка аппаратной части установки	Эксперт Исполнитель	16	
4	Разработка архитектуры программного обеспечения	Исполнитель	3	
5	Разработка загрузчика микроконтроллера	Эксперт Исполнитель	10	
6	Разработка основного программного обеспечения микроконтроллера	Исполнитель	12	
7	Разработка протоколов взаимодействия	Исполнитель	12	
8	Разработка программного обеспечения ПК	Исполнитель	16	
9	Проведение экспериментов с установкой обезвешивания	Исполнитель	6	
10	Оценка полученных результатов	Руководитель Эксперт Исполнитель	3	

Цветовое обозначение – Руководитель – Эксперт – Исполнитель

3.3.2 Бюджет научного исследования

Рассмотрим 2 варианта исполнения научно-технического исследования:

1) текущая магистерская диссертация,
2) решение с использованием средств блочного программирования

В таблице 3.13 представлена группировка затрат по статьям.

Таблица 3.13 – Группировка затрат по статьям

	Статьи			
Варианты исполнения	Сырьё, материалы, покупные изделия и полуфабрикаты, руб.	Специальное оборудование для научных (экспериментальных) работ, руб.	Основная заработная плата с отчислениями в социальные внебюджетные фонды, руб.	Итого плановая стоимость, руб.
1	2 885,38	507 407,63	346 320	856 613,01
2	2 885,38	443 866,70	317 460	764 212,08

В статью сырьё, материалы, комплектующие изделия и покупные полуфабрикаты отнесём электроэнергию и сеть Интернет. Примем состав этой статьи расходов одинаков для всех видов исполнения проекта. В таблице 3.14 приведён расчёт стоимости используемых ресурсов.

Таблица 3.14 – Сырьё, материалы, комплектующие изделия и покупные полуфабрикаты

Наименование	Цена за месяц/киловатт-час, руб.	Количество месяцев/киловатт-часов	Сумма, руб.
Сеть Интернет 100 Мб/с	600,00	4	2 400,00
Электроэнергия	3,16	153,6	485,38
		Итог	2 885,38

В статью специальное оборудование для научных работ также включим различное программного обеспечение, используемое для реализации проекта. В таблицах 3.15, 3.16 приведены расчёты затрат по статье спецоборудования для первого, второго и третьего вариантов исполнения, соответственно.

Таблица 3.15 – Расчёт затрат по статье спецоборудование для научных работ для первого варианта исполнения проекта

№ п/п	Наименование оборудования и программного обеспечения	Общая стоимость, руб.
1	Операционная система Microsoft Windows 10 Home	17 699,00
2	Microsoft Office для дома и бизнеса	29 990,00
3	Пакет прикладных программ Matlab (годовая подписка)	74 660,59
4	Среда блочного программирования Simulink (годовая подписка)	112 785,15
5	Control System Toolbox для Matlab (годовая подписка)	42 890,13
6	Qt Creator (годовая подписка)	37 171,44
7	System Identification Toolbox для Matlab (годовая подписка)	42 890,13
8	Optimization Toolbox для Matlab (годовая подписка)	42 890,13
9	Simulink Design Optimization для Simulink (годовая подписка)	42 890,13
10	Econometrics Toolbox для Matlab (годовая подписка)	63 540,93
	Итог	507 407,63

Таблица 3.16 – Расчёт затрат по статье спецоборудование для научных работ для второго варианта исполнения проекта

№ п/п	Наименование оборудования и программного обеспечения	Общая стоимость, руб.
1	Операционная система Microsoft Windows 10 Home	17 699,00
2	Microsoft Office для дома и бизнеса	29 990,00
3	Пакет прикладных программ Matlab (годовая подписка)	74 660,59
4	Среда блочного программирования Simulink (годовая подписка)	112 785,15
5	Control System Toolbox для Matlab (годовая подписка)	42 890,13
6	Qt Creator (годовая подписка)	37 171,44
7	System Identification Toolbox для Matlab (годовая подписка)	42890,13
8	Optimization Toolbox для Matlab (годовая подписка)	42 890,13
9	Simulink Design Optimization для Simulink (годовая подписка)	42 890,13
	Итог	443 866,70

В завершении формирования бюджета проекта рассчитаем основную заработную плату и отчисления на социальные нужды. Расчёт для первого варианта исполнения проекта приведён в таблице 3.17, для второго варианта – в таблице 3.18.

Таблица 3.17 – Расчёт основной заработной платы и отчисления на социальные нужны для первого варианта исполнения

Сотрудник	Трудоёмкость, чел.-час	Тариф с районным коэффициентом, руб. за 1 чел./час	Всего заработная плата, руб.	Отчисления во внебюджетные фонды	Итог, руб.
Руководитель	20	600	12 000	79 920	346 320
Эксперт	40	600	24 000		
Исполнитель	768	300	230 400		

Таблица 3.18 – Расчёт основной заработной платы и отчисления на социальные нужны для второго варианта исполнения

Сотрудник	Трудоёмкость, чел.-час	Тариф с районным коэффициентом, руб. за 1 чел./час	Всего заработная плата, руб.	Отчисления во внебюджетные фонды	Итог, руб.
Руководитель	45	600	27 000	73 260	317 460
Эксперт	80	600	48 000		
Исполнитель	564	300	169 200		

3.4 Определение ресурсной, финансовой, бюджетной, социальной и экономической эффективности исследования

3.4.1 Оценка сравнительной эффективности исследования

Начнём с оценки финансовой эффективности научного исследования. Для этого посчитаем интегральный финансовый показатель для двух видов исполнения научно-технического проекта

$$I_{ф}^{p} = \frac{856\,613,01}{856613,01} = 1,$$

$$I_{ф}^{a1} = \frac{764\,212,08}{856613,01} = 0,89$$

Теперь приступим к оценке ресурсоэффективности вариантов исполнения проекта. Для этого составим таблицу 3.19, в которой оценим варианты исполнения проекта по общему набору критериев. Критерии взяты из анализа конкурентных решений.

Таблица 3.19 – Сравнительная оценка характеристик вариантов исполнения проекта

Критерии оценки	Вес критерия	Баллы	
		текущего проекта	первого аналога
1. Требования к вычислительным ресурсам	0,2	4	4
2. Помехоустойчивость	0,2	3	2
3. Точность оценки величины проскальзывания колёс	0,3	4	3
4. Универсальность системы	0,3	4	3
Итого	**1,0**		

По данным таблицы 3.19 вычислим интегральный показатель ресурсоэффективности вариантов исполнения научного-технического проекта

$$I_{m}^{p} = 0,2 * 4 + 0,2 * 3 + 0,3 * 4 + 0,3 * 4 = 3,8,$$

$$I_{m}^{a1} = 0,2 * 4 + 0,2 * 2 + 0,3 * 3 + 0,3 * 3 = 3,0.$$

Переходим к подсчёту интегральных показателей эффективности разработки и аналогов

$$I_{финр}^{p} = \frac{I_{m}^{p}}{I_{ф}^{p}} = \frac{3,8}{1} = 3,8,$$

$$I_{финр}^{a1} = \frac{I_{m}^{a1}}{I_{ф}^{a1}} = \frac{3,0}{0,89} = 3,4.$$

Последним показателем, который необходимо вычислить, является сравнительная эффективность проекта

$$Э_{ср} = \frac{I_{финр}^{p}}{I_{финр}^{a1}} = \frac{3,8}{3,4} = 1,11.$$

В заключении проведём сравнение финансовой и ресурсной эффективности вариантов исполнения в целом. В таблице 3.20 сведены все показатели эффективности.

Таблица 3.20 – Сравнительная эффективность разработки

№ п/п	Показатель	Аналог 1	Разработка
1	Интегральный финансовый показатель разработки	0,89	1
2	Интегральный показатель ресурсоэффективности разработки	3,00	3,80
3	Интегральный показатель эффективности	3,4	3,8
4	Сравнительная эффективность вариантов исполнения		1,11

Итак, финансовая эффективность исполнений проекта показала, что затраты меньше у первого аналога разработки на 11%. Но по интегральному показателю ресурсоэффективности первый аналог уступает текущему варианту технического решения. Сравнительная оценка эффективности вариантов исполнения нашей разработки показала, что текущее исполнение превосходит на 11% второй вариант исполнения. За счет того, что текущий вариант исследования предполагает использования долее дорогих ресурсов для обеспечения точности оценки величины проскальзывания колес, универсальности системы и помехоустойчивости.

Заключение

В ходе выполнения выпускной квалификационной работы был выполнен ряд задач:

1) Доработана аппаратная часть существующей системы – добавлен микроконтроллер, а также платы модулей датчиков и внешней памяти;
2) Разработана архитектура комплекса программного обеспечения, включающая в себя программное обеспечение микроконтроллера, а также программное обеспечение персонального компьютера;
3) Был разработан протокол обмена данными между микроконтроллером и персональным компьютером;
4) Было разработано программное обеспечение микроконтроллера, включающее в себя загрузчик и программу для управления двигателями;
5) Было разработано основное программное обеспечение персонального компьютера, в которое вошло три модуля: модуль загрузчика, модуль управления перемещением объекта обезвешивания, а также экспериментальный модуль блочного программирования;
6) Было разработано вспомогательное программное обеспечение персонального компьютера, в которое вошли две программы: программа отслеживания положения объекта обезвешивания по ArUco маркеру и программа построения сложных траекторий для пространственного перемещения объекта обезвешивания;
7) На базе всего программно-аппаратного комплекса был проведен эксперимент по пространственному перемещению объекта по горизонтальной окружности. В ходе эксперимента были выявлены аппаратные недостатки и недостатки системы оценки положения объекта по математическим моделям.

По итогу выполнения ВКР была разработана система обезвешивания, которая применима в учебно-исследовательских целях, а также, при доработке, может быть применена и в производственной сфере.

Список используемых источников

1. Aretech – Robotic Body-Weight Support Systems. – URL: https://www.aretechllc.com/ (date accessed: 05.06.2023). – Text: electronic.
2. Vector – VECTOR Advantage. – URL: https://go.bioness.com/Vector-PTOT (date accessed: 05.06.2023). – Text: electronic.
3. REHA STIM – The FLOAT. – URL: https://reha-stim.com/the-float_2/ (date accessed: 05.06.2023). – Text: electronic.
4. Motek – RYSEN. – URL: https://www.motekmedical.com/solution/rysen/ (date accessed: 05.06.2023). – Text: electronic.
5. ST – STM32 32-bit Arm Cortex MCUs. – URL: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html (date accessed: 05.06.2023). – Text: electronic.
6. 17 Атрибутов Хорошего Канального Протокола Передачи Данных – Хабр. – URL: https://habr.com/ru/articles/682292/ (дата обращения: 05.06.2023). – Текст: электронный.
7. Real-time operating system for microcontrollers – FreeRTOS. – URL: https://www.freertos.org/ (date accessed: 05.06.2023). – Text: electronic.
8. Как сделать context switch на STM32. – URL: https://habr.com/ru/companies/embox/articles/330236/ (дата обращения: 05.06.2023). – Текст: электронный.
9. Programming Robotino View – FESTO. – URL: https://ip.festo-didactic.com/InfoPortal/Robotino/Software/Programming/EN/RobotinoView.html (date accessed: 05.06.2023). – Text: electronic.
10. СанПиН 2.2.2/2.4.1340-03. Гигиенические требования к персональным электронно-вычислительным машинам и организации работы : дата введения 2003-06-30. – Москва : Федеральный центр госсанэпиднадзора Минздрава России, 2003.
11. СНиП 2.04.05-86. Отопление, вентиляция и кондиционирование : дата введения 1986-01-01. – Москва : ЦИТП Госстроя СССР, 1987.

12. ГН 2.2.5.1313-03. Предельно допустимые концентрации (ПДК) вредных веществ в воздухе рабочей зоны : дата введения 2003-06-15. – Гигиенические нормативы ГН 2.2.5.1313-03.
13. ОСТ 54 30013-83. ССБТ. Электромагнитные излучения СВЧ. Предельно допустимые уровни облучения. Требования безопасности : дата введения 1984-01-01. – Москва.
14. ГОСТ 12.1.004-91. Система стандартов безопасности труда (ССБТ). Пожарная безопасность. Общие требования (с Изменением N 1) : дата введения 1992-07-01. – Москва : Стандартинформ, 2006.
15. ГОСТ 12.1.010-76. Система стандартов безопасности труда (ССБТ). Взрывобезопасность. Общие требования (с Изменением N 1) : дата введения 1978-01-01. – Москва : ИПК Издательство стандартов, 2002.
16. ГОСТ Р 12.1.019-2009. Система стандартов безопасности труда (ССБТ). Электробезопасность. Общие требования и номенклатура видов защиты : дата введения 2009-12-10. – Москва : Стандартинформ, 2010.
17. ГОСТ 12.1.030-81. Система стандартов безопасности труда (ССБТ). Электробезопасность. Защитное заземление. Зануление (с Изменением N 1) : дата введения 1982-07-01. – Москва : ИПК Издательство стандартов, 2001.
18. НПБ 105-03. Определение категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности : дата введения 2003-08-01. – Москва : ФГУ ВНИИПО МЧС России, 2003.
19. СНиП 2.01.02-85. Противопожарные нормы : дата введения 1987-01-01. – Москва : Госстрой СССР, 1991.
20. ГОСТ 12.4.021-75. Система стандартов безопасности труда (ССБТ). Системы вентиляционные. Общие требования (с Изменением N 1) : дата введения 1977-01-01. – Москва : ИПК Издательство стандартов, 2001.
21. ГОСТ 9230-77. Огнетушители СО(2) (углекислотные) передвижные. Технические условия (с Изменением N 1) : дата введения 1978-01-01. – Москва : Издательство стандартов, 1983.

22. ТУ 22-4720-80. Огнетушитель химический воздушно-пенный ОХВП-10 : дата введения 1981-01-01.

Приложение А
(справочное)

Раздел 1.1
Обзор существующих решений

Обучающийся:

Группа	ФИО	Подпись	Дата
8ЕМ11	Иванов Егор Андреевич		

Консультант ОАР ИШИТР:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Заведующий кафедрой – руководитель отделения на правах кафедры ОАР ИШИТР	Филипас А.А.	к.т.н., доцент		

Консультант – лингвист ОИЯ ШБИП:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент (ОИЯ)	Диденко А.В.	к.ф.н., доцент		

1.1 Overview of existing solutions

Today, cable weight compensation systems are actively used in various fields. These systems are quite widespread in the medical industry. There, these systems are used to rehabilitate patients with musculoskeletal injuries. The approach to the development of such systems is similar, but there are some differences. Below some of the cable-operated weightlessness systems are demonstrated.

One of such systems is the Aretech ZeroG (Fig. 1.1). This rehabilitative weight compensation system is based on a single link arm. With a single cable attached to a ceiling-mounted rail, this system allows the patient to move in a predetermined area.

Figure 1.1 – Aretech ZeroG

Another similar system is the Bioness Vector (Fig. 1.2). In its essence and design it is very similar to the ZeroG system. This system also has a rail along which the carriage with a cable attached to it moves.

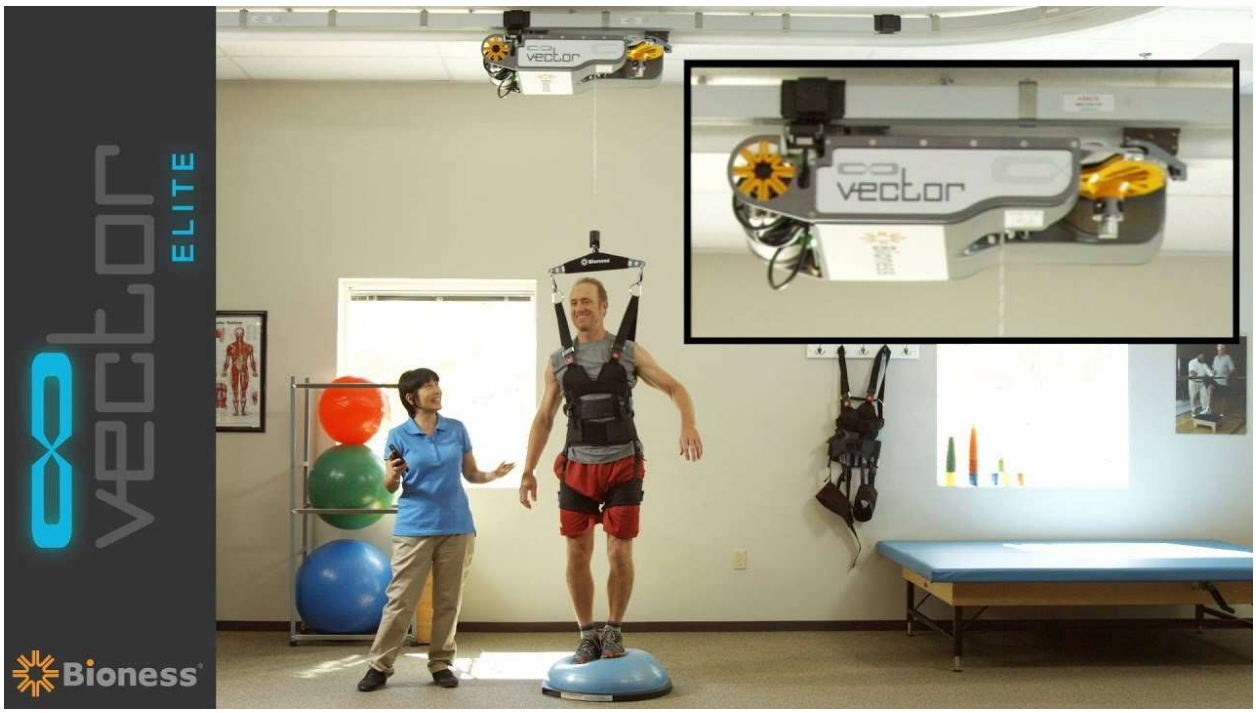


Figure 1.2 – Bioness Vector

Float is another cable weight compensation system (Figure 1.3). This system, unlike the above mentioned, includes four flexible cables. These cables are connected to moving rollers which move along parallel rails attached to the ceiling. The rollers are moved by actuators.

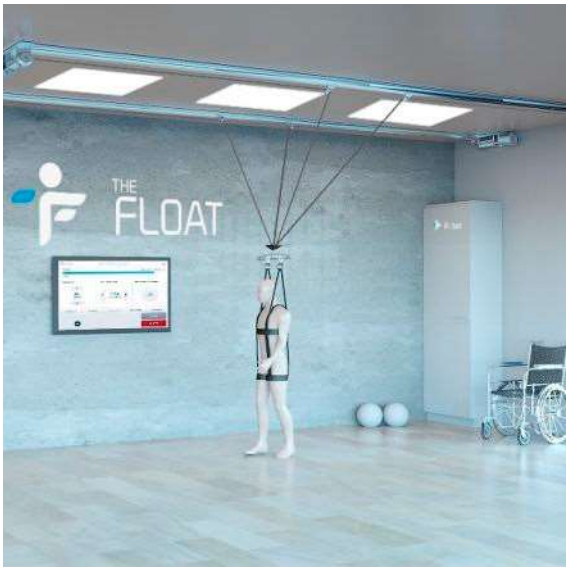


Figure 1.3 – Float

There is also the Ryzen system from Motekmedikal (Figure 1.4). This system is similar to the Float system. It also has four cables attached through rollers that are moved along parallel ceiling-mounted rails by actuators.

Figure 1.4 – Motekmedikal Ryzen

All of the above systems are the most popular cable weight compensation systems currently used in medicine.

The disadvantages of the first two systems are the significant travel limitations, since the object has to move strictly under the rail for correct de-weighting.

The last two systems have a small disadvantage, which is the fixed length of the cable. Thus, the active area is limited directly by the height of the object - the higher the object is, the more distance between the two cables on one rail and the smaller the active travel area of the whole system will be.