МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования
«Гомельский государственный технический университет имени П.О. Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Специальность 1-40 05 01 «Информационные системы и технологии (по направлениям)»

Направление специальности 1-40 05 01-01 «Информационные системы и технологии (в проектировании и производстве)»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломной работе
на тему «Программные средства обеспечения движения аграрного робота по заданной траектории»

Разработал студент гр. ИТП-41 _____ Расшивалов Н.И.
(подпись) (Ф.И.О.)

Руководитель работы _____ доцент, к.т.н. Курочка К.С.
(подпись) (учёное звание, учёная степень, Ф.И.О.)

Консультант по экономической части _____ доцент, к.э.н. Громыко Р.И.
(подпись) (учёное звание, учёная степень, Ф.И.О.)

Консультант по охране труда и ресурсо-энергосбережению _____ доцент, к.т.н. Токочаков В.И.
(подпись) (учёное звание, учёная степень, Ф.И.О.)

Нормоконтроль _____ доцент, к.т.н. Токочаков В.И.
(подпись) (учёное звание, учёная степень, Ф.И.О.)

Рецензент _____ _____
(подпись) (учёное звание, учёная степень, должность, организация Ф.И.О.)

Дипломная работа (_____с.) допущена к защите в Государственной экзаменационной комиссии.

Зав. кафедрой
«Информационные технологии» _____ доцент, к.т.н. Курочка К.С.
(подпись) (учёное звание, учёная степень, Ф.И.О.)

Гомель 2023

Реферат

ПРОГРАММНЫЕ СРЕДСТВА ОБЕСПЕЧЕНИЯ ДВИЖЕНИЯ АГРАРНОГО РОБОТА ПО ЗАДАННОЙ ТРАЕКТОРИИ: дипломная работа / Н. И. Расшивалов. – Гомель: ГГТУ им. П.О. Сухого, 2023. – Дипломная работа: 100 страниц, 16 рисунков, 12 таблиц, 12 источников, шесть приложений.

Ключевые слова: роботы, навигация, одноплатные компьютеры, сетевое взаимодействие, управление.

Объектом разработки является система навигации для колесного робота.

Целью работы является создание программных средств обеспечения движения колесного робота по заданной траектории.

Этапы выполнения работы включают в себя:

– аналитический обзор колесных роботов;

– аналитический обзор способов навигации колесных роботов;

– аналитический обзор средств формирования спутниковых карт;

– аналитический обзор средств формирования скриптов для управления колесными роботами;

– проектирование и разработка программных средств;

– произведения экономического расчёта и обоснования рентабельности разработки;

– анализа внедрения разработки с точки зрения вопросов энерго- и ресурсосбережения.

Студент-дипломник подтверждает, что дипломная работа выполнена самостоятельно, приведенный в дипломной работе материал объективно отражает состояние разрабатываемого объекта, пояснительная записка проверена в системе «Антиплагиат» (*https*://*www*.*antiplagiat*.*ru*/). Степень оригинальности текста составляет 89,51 %. Все заимствованные из литературных и других источников, теоретические и методологические положения и концепции сопровождаются ссылками на источники, указанные в вписке использованных источников.

Резюме

Тема дипломной работы: «Программные средства обеспечения движения аграрного робота по заданной траектории».

Объект исследования: системы искусственного интеллекта для колёсных аграрных роботов.

Предмет исследования: система автоматизированной навигации для аграрного колесного робота.

Цель работы: разработка программного обеспечения, управляющего движением аграрного колесного робота по заданной траектории.

Основной результат работы: программный продукт для одноплатного компьютера, реализующий скрипт управления движением робота.

Рэзюмэ

Тэма дыпломнай працы: «Праграмныя сродкі забеспячэння руху аграрнага робата па зададзенай траекторыі».

Аб'ект даследавання: сістэмы штучнага інтэлекту для колавых аграрных робатаў.

Прадмет даследавання: сістэма аўтаматызаванай навігацыі для аграрнага колавага робата.

Мэта працы: распрацоўка праграмнага забеспячэння, кіраўніка рухам аграрнага колавага робата па зададзенай траекторыі.

Асноўны вынік працы: праграмны прадукт для аднаплатнага кампутара, які рэалізуе скрыпт кіравання рухам робата.

Abstract

The topic of the thesis: «Software tools for ensuring the movement of an agricultural robot along a given trajectory».

Object of research: artificial intelligence systems for wheeled agricultural robots.

Subject of research: automated navigation system for agricultural wheeled robot.

Purpose of the work: development of software that controls the movement of an agricultural wheeled robot along a given trajectory.

The main result of the work: a software product for a single-board computer that implements a robot motion control script.

ОГЛАВЛЕНИЕ

Перечень условных обозначений и сокращений

В настоящей пояснительной записке применяются следующие термины, обозначения и сокращения.

АКБ – аккумуляторная батарея.

ВЛ – воздушные линии.

ЗУ – заземляющее устройство.

ИИ – искусственный интеллект.

МО – машинное обучение.

ОЗУ – оперативное запоминающее устройство.

ПК – персональный компьютер.

ПО – программное обеспечение.

ПП – программный продукт.

ПУЭ – правила устройства электроустановок.

ШИМ – широтно-импульсная модуляциия.

ЭВМ – электронно-вычислительная машина.

CORS (*Cross-Origin Resource Sharing*) – совместное использование ресурсов между источниками.

GPS (Global Positioning System) – система глобального позиционирования

IMU (*Inertial Measurement Unit*) – инерциальные измерительные блоки

IoT (Internet of Things) – термин описание интернет вещей.

SLAM (*Simultaneous Localization and Mapping*) – одновременное построение карты и определение местоположения.

VNC (*Virtual Network Computing*) – система удаленного доступа.

UART (Universal Asynchronous Receiver-Transmitter) – узел вычислительных устройств, предназначенный для организации.

ВВЕДЕНИЕ

Программные средства обеспечения движения колесного робота по заданной траектории являются важными инструментами для автоматизации планирования маршрутов колесных роботов. Они позволяют пользователям быстро и легко создавать маршруты на основе данных спутниковой карты, а также передавать их роботу для последующего выполнения.

Кроме того, для эффективного выполнения задач автономными колесными роботами необходимо разработать программные средства обеспечения движения по заданной траектории. Эти средства позволят роботам следовать траектории и выполнять задачи в различных областях, таких как сельское хозяйство, логистика и транспорт.

Навигация является важным аспектом автономного функционирования колесных роботов. Программные средства обеспечения движения колесного робота по заданной траектории обеспечивают эффективное планирование маршрутов и управление движением роботов.

Колесный робот может функционировать под управлением пользователя, например, с пульта, или работать автономно. Благодаря использованию спутниковых карт и программных средств навигации, роботы могут эффективно планировать свои перемещения, избегать препятствий и доставлять товары или выполнять другие задачи с минимальным участием человека. Это способствует автоматизации процессов и повышению эффективности функционирования роботов в различных отраслях промышленности и обслуживания.

Программные средства обеспечения движения колесного робота по заданной траектории должны включать в себя следующие функциональные возможности:

– планирование траектории: разработка алгоритмов для определения оптимальной траектории движения робота по заданным точкам или областям на спутниковой карте;

– управление движением: разработка алгоритмов для управления колесным роботом на основе заданной траектории.

Актуальность заключается в том, что разработка такого вида программного обеспечения может снизить затраты на трудовые ресурсы и эксплуатацию техники, а также позволит роботам выполнять свои функции в условиях, где возможности человека ограничены.

1 АППАРАТНО-ПРОГРАММНЫЕ СРЕДСТВА ОБЕСПЕЧЕНИЯ ДВИЖЕНИЯ АГРАРНОГО РОБОТА ПО ЗАДАННОЙ ТРАЕКТОРИИ

1.1 Распространение колесных роботов и способы их навигации

Распространение колесных роботов в сельском хозяйстве и других отраслях становится все более значимым с каждым годом. Эти автономные устройства представляют собой передовую технологию, способную эффективно улучшить производительность и управление в сельскохозяйственных операциях, а также в других областях промышленности.

Одним из основных преимуществ колесных роботов является их способность выполнять множество задач без необходимости прямого участия человека.

При помощи использования различных датчиков и систем навигации, эти роботы способны автономно перемещаться по территории, определять свое местоположение и преодолевать преграды.

В сельском хозяйстве колесные роботы активно применяются для выполнения различных задач. Они могут использоваться для автоматического полива и удобрения посевов, сбора урожая, обработки почвы и многих других процессов.

Благодаря своей автономности и точности, роботы способны повысить эффективность и уровень продуктивности в сельском хозяйстве, сократить затраты на рабочую силу и ресурсы.

Однако, распространение колесных роботов в сельском хозяйстве требует ряда технических улучшений и инноваций. Одной из ключевых задач является разработка более точных систем навигации и определения положения, чтобы роботы могли более точно выполнять свои задачи и избегать повреждений посевов или препятствий.

Кроме того, автоматизация колесных роботов требует разработки специализированных алгоритмов и программного обеспечения. Это позволит роботам выполнять сложные задачи, такие как распознавание и сортировка продукции, контроль состояния растений и принятие решений, на основе собранных данных.

Колесные роботы играют важную роль в различных отраслях, особенно в сельском хозяйстве, и вносят значительный вклад в современное развитие сельскохозяйственного производства.

Они предлагают новые возможности для повышения эффективности, улучшения качества продукции и оптимизации процессов. Чтобы полностью реализовать потенциал колесных роботов, необходимы дальнейшие технические улучшения и интеграция существующих систем. Навигация колесных роботов основана на нескольких компонентах, которые работают вместе для определения

и контроля положения робота. Это сложная система, позволяющая роботу ориентироваться в окружающей среде и перемещаться по ней.

Одним из ключевых компонентов навигационной системы являются датчики измерения расстояния. Робот обычно оснащен различными типами датчиков, такими как ультразвуковые, инфракрасные или лазерные датчики. Они позволяют роботу определять расстояние до препятствий и стен, что помогает избегать столкновений.

Другим важным компонентом навигации является система позиционирования. Она позволяет роботу определять свое текущее положение в пространстве. Для этого используются различные технологии, включая глобальную позиционную систему, инерциальные измерительные блоки и компасы. Эти компоненты предоставляют роботу информацию о его координатах и ориентации.

Контроль движения является еще одним важным аспектом навигации колесных роботов. Он включает систему управления моторами колес и других приводов. Робот может использовать эту систему для контроля скорости, направления и поворотов. Он принимает решения о том, какое колесо или группа колес должна вращаться и с какой скоростью, чтобы достичь требуемого перемещения.

Взаимодействие всех этих компонентов осуществляется через центральный контроллер или компьютер, который обрабатывает данные с датчиков, принимает решения о дальнейшем перемещении и управляет движением робота.

Кроме описанных компонентов, существуют другие виды навигации и технические средства для их реализации. Например, стерео-зрение позволяет роботам видеть и анализировать окружающую среду в трехмерном формате, определять расстояния до объектов, их форму и размеры, а также обнаруживать препятствия.

Сенсоры силы и касания помогают роботам определить, когда они сталкиваются с препятствием или прикладывают силу к объекту. Полученные данные используются для корректировки траектории движения робота или для взаимодействия с окружающими объектами.

Для более точной навигации роботы могут использовать системы составления карт и локализации. Они позволяют роботу построить карту окружающей среды и определить свое местоположение на ней. Для этого могут использоваться данные от лазерных сканеров или камер в сочетании с алгоритмами *SLAM*.

Роботы также могут использовать маркеры или маяки, размещенные в окружающей среде, для определения своего положения. Эти маркеры могут быть визуальными, например, цветными, или использовать другие технологии такие, как радиочастотную идентификацию. Роботы обнаруживают эти маркеры с помощью своих датчиков и используют их для ориентации и навигации.

Техники искусственного интеллекта и машинного обучения также могут быть применены для улучшения навигации колесных роботов. Нейронные сети

и алгоритмы обучения с подкреплением позволяют роботам принимать решения на основе данных с датчиков и опыта, что помогает им адаптироваться к различным ситуациям и условиям окружающей среды.

Все эти технические средства могут быть комбинированы и интегрированы в навигационную систему колесных роботов, обеспечивая им более точную и надежную способность перемещения и ориентации в окружающей среде.

1.2 Архитектура колесного робота

Архитектура колесного робота включает в себя следующие основные компоненты:

– колесная система – колеса являются главными элементами для передвижения колесного робота. Они обеспечивают подвижность и маневренность робота. Колеса могут быть различных типов, например, обычные колеса, ролики или гусеницы;

– погонная система – состоит из двигателей, передаточных механизмов (например, зубчатых передач, редукторов) и осей, которые передают движение от двигателей к колесам. Погонная система отвечает за передвижение робота;

– сенсоры используются для получения информации о внешней среде и состоянии робота. Примеры сенсоров включают датчики расстояния, инфракрасные сенсоры, ультразвуковые сенсоры, гироскопы, акселерометры, камеры. Сенсоры предоставляют данные для принятия решений и управления роботом;

– управляющая система – обрабатывает данные от сенсоров и принимает решения о действиях робота. Это может быть компьютерное устройство или микроконтроллер, которое выполняет алгоритмы управления и управляет двигателями и другими устройствами робота;

– источник питания – колесные роботы обычно питаются от аккумуляторов или других источников энергии. Источник питания обеспечивает электроэнергию для работы всех компонентов робота.

Системы управления движением колесных роботами могут быть реализованы с использованием различных подходов, методов и языков программирования.

Классификации языков управления роботами:

– низкоуровневые языки – включают языки, такие как Ассемблер и *C*, которые позволяют непосредственное управление аппаратными компонентами робота. Они обычно используются для написания кода, управляющего двигателями и обрабатывающего сенсорные данные;

– высокоуровневые языки – включают языки, такие как *Python*, *MATLAB*. Они предоставляют более абстрактный уровень и позволяют разработчикам

легко программировать функциональность робота, такую как планирование движения, обработка изображений или общение с другими роботами;

– графические среды программирования – предоставляют визуальные средства для программирования робота без необходимости написания кода такие как *Blockly*, *Scratch* или *LabVIEW*.

Использование методов машинного обучения, таких как нейронные сети или алгоритмы обучения с подкреплением, для обучения робота оптимальным стратегиям управления на основе данных из сенсоров и обратной связи.

Применение классических алгоритмов управления, таких как ПИД-регуляторы – пропорционально-интегрально-дифференциальные регуляторы, для управления движением робота на основе измерений сенсоров.

Использование алгоритмов планирования движения для определения оптимального пути или траектории робота с учетом ограничений окружающей среды и целей задачи.

Колесные роботы обладают коммуникационными возможностями для обмена данными с внешними устройствами или другими роботами. Они могут включать протоколы беспроводной связи, такие как *Wi-Fi*, *Bluetooth* или радиочастотная связь.

1.3 Сравнительная характеристика одноплатных компьютеров для управления движением робота

Для управления движением робота могут быть использованы следующие одноплатные компьютеры:

– *Asus Tinker Board;*
– *Raspberry Pi;*
– *Orange Pi;*
– *Banana Pi.*

В таблице 1.1 находятся основные характеристики одноплатных компьютеров.

Таблица 1.1 – Сравнительная характеристика одноплатных компьютеров

Основные характеристики	*Asus Tinker Board*	*Raspberry Pi*	*Orange Pi*	*Banana Pi*
1	2	3	4	5
Процессор	*Rockchip RK*3288 с 4 ядрами	*Allwinner H*3/*H*5/*H*6 на выбор	*Broadcom BCM*2835/ *BCM*2711	*Allwinner A*20/*A*64/*R*40 на выбор
Графический процессор	*ARM Mali-T*764 *GPU*	*Mali*-400 *MP*2/*MP*4	*Broadcom VideoCore*	*Mali*-400 *MP*2/*MP*4

Продолжение таблицы 1.1

1	2	3	4	5
Оперативная память	2 Гб *DDR*3 *RAM*	512 МБ, 1 Гб, 2 Гб или 4 Гб *DDR*3 *RAM*	от 1 Гб до 8 Гб *LPDDR*4-3200 *SDRAM* на модели *Raspberry Pi* 4	512 МБ, 1 Гб, 2 Гб или 4 Гб *DDR*3 *RAM*
Сеть	*Gigabit Ethernet*, 802.11 *b/g/n Wi-Fi* и *Bluetooth* 4.0	*10*/*100M Ethernet*, 802.11 *b/g/n Wi-Fi* и *Bluetooth* 4.0 на некоторых моделях	*Gigabit Ethernet*, 802.11 *b/g/n/ac Wi-Fi* и *Bluetooth* 4.2/*BLE* на некоторых моделях	10/100/1000M *Ethernet*, 802.11 *b/g/n Wi-Fi* и *Bluetooth* 4.0 на некоторых моделях
Операционная система	поддерживает различные операционные системы, включая *Debian*, *Ubuntu* и *Android*	поддерживает различные операционные системы, включая *Ubuntu*, *Debian*, *Android*, *Armbian* и другие	поддерживает различные операционные системы, включая *Raspbian*, *Ubuntu*, *Debian*, *Android*, *Windows* 10	поддерживает различные операционные системы, включая *Ubuntu*, *Debian*, *Android*,

Asus Tinker Board – это компактный одноплатный компьютер, разработанный для различных проектов, требующих высокой производительности и графической мощности. Он предлагает широкие возможности в области мультимедиа, интернета вещей (*IoT*), робототехники и других приложений.

Одна из ключевых особенностей *Asus Tinker Board* – это его производительность. Он оснащен четырех-ядерным процессором *Rockchip RK*3288 с тактовой частотой до одной целой восьми десятых ГГц, что позволяет выполнять разнообразные задачи быстро и эффективно. Вместе с графическим процессором *Mali-T*764 обеспечивается поддержка высококачественной графики и видео.

Платформа *Asus Tinker Board* предлагает разнообразные порты и интерфейсы для подключения различных периферийных устройств и расширений [1].

Включает в себя *HDMI*-порт для подключения к монитору или телевизору, четыре *USB*-порта для подключения устройств хранения данных, клавиатуры, мыши и других аксессуаров, а также разъемы *Ethernet*, аудио и *GPIO* для обеспечения связи с внешними устройствами.

Asus Tinker Board работает на операционной системе *TinkerOS*, которая основана на *Debian Linux*. Это позволяет разработчикам использовать множество программных инструментов и библиотек для создания и настройки своих проектов.

Благодаря своей мощности и гибкости, *Asus Tinker Board* подходит для широкого спектра применений. Он может использоваться его для создания мультимедийных систем, таких как медиацентры или стримеры, разрабатывать проекты *IoT*, например, умные дома или устройства сбора данных, а также встраивать его в робототехнические системы для обработки данных и управления.

Orange Pi – это семейство одноплатных компьютеров, представляющих собой компактные и экономичные устройства. Они разработаны с использованием открытых технологий и основаны на *ARM*-процессорах, что обеспечивает хорошую производительность при низком энергопотреблении.

Одна из основных особенностей *Orange Pi* – его многообразие моделей и конфигураций. В зависимости от конкретной модели, устройства могут быть оснащены различными процессорами (например, *ARM Cortex-A*7, *Cortex-A*53 или *Cortex-A*64), объемом оперативной памяти, встроенной памяти и другими параметрами.

Orange Pi поддерживает различные операционные системы, включая *Linux* (*Ubuntu*, *Debian* и другие дистрибутивы) и *Android*. Это делает его гибким и удобным инструментом для разработчиков и энтузиастов, которые хотят создавать собственные проекты в области *IoT* домашних медиацентров, систем контроля доступа и многих других областях.

Благодаря своим компактным размерам и разнообразию интерфейсов (*USB*, *Ethernet*, *HDMI*, аудио и другие), *Orange Pi* легко интегрируется в различные устройства и системы. Он может использоваться как основа для создания множества проектов, включая умные дома, автоматизированные системы, робототехнику, серверы и другие приложения.

Orange Pi – это доступное и гибкое решение, сочетающее в себе компактность и достаточную производительность для широкого спектра задач.

Raspberry Pi – это одноплатный компьютер, который предлагает различные модели и конфигурации для различных проектов. Он стал популярным инструментом для создания *IoT*-устройств, медиацентров, систем видеонаблюдения и других приложений, которые не требуют высокой вычислительной мощности. У *Raspberry Pi* также есть различные варианты оперативной памяти, в зависимости от модели. Базовые модели *Raspberry Pi* имеют один Гб ОЗУ, но более

продвинутые модели могут предлагать два Гб, четыре Гб, восемь Гб и даже 16 Гб ОЗУ. Больший объем оперативной памяти обеспечивает более плавную работу и позволяет запускать более ресурсоемкие задачи.

Хранилище данных на *Raspberry Pi* может быть оснащено различными опциями. Базовые модели *Raspberry Pi* используют карты памяти *microSD* для хранения операционной системы и данных. Однако можно также подключить внешние накопители данных, такие как *USB*-накопители или жесткие диски, чтобы расширить доступное хранилище.

В сетевых возможностях *Raspberry Pi* также предоставляются различные варианты. Большинство моделей *Raspberry Pi* имеют встроенные порты *Ethernet*, что обеспечивает проводное подключение к сети. Кроме того, многие модели *Raspberry Pi* также поддерживают *Wi-Fi* и *Bluetooth*, что позволяет беспроводное подключение к сети и другим устройствам.

Raspberry Pi работает на различных операционных системах, основанных на *Linux*, таких как *Raspbian*, *Ubuntu* и другие. Он также поддерживает *Windows* 10 *IoT Core*.

Raspberry Pi имеет огромное сообщество разработчиков, что обеспечивает обширную поддержку, множество проектов и обмен опытом. Есть широкий выбор дополнительных модулей и аксессуаров, таких как дисплеи, камеры, сенсоры и расширительные платы *GPIO*, которые расширяют возможности *Raspberry Pi* и позволяют создавать уникальные проекты.

Raspberry Pi является мощным и гибким одноплатным компьютером, который позволяет создавать различные проекты с разными требованиями.

Banana Pi – это семейство одноплатных компьютеров, предназначенных для различных проектов, аналогичных *Raspberry Pi*.

Оперативная память на *Banana Pi* может варьироваться в зависимости от конкретной модели и конфигурации. Некоторые модели имеют один Гб ОЗУ, в то время как другие могут иметь два Гб, четыре Гб или даже больше. Большой объем ОЗУ позволяет обрабатывать более сложные задачи и запускать более требовательные приложения.

Хранилище данных на *Banana Pi* разнообразно. Он может быть оснащен встроенной флеш-памятью, которая обычно варьируется от нескольких гигабайт до нескольких десятков гигабайт.

Кроме того, на *Banana Pi* можно подключить внешние накопители данных, такие как жесткие диски или карты памяти, чтобы расширить доступное хранилище.

Сетевые возможности *Banana Pi* – он может быть оснащен различными интерфейсами, такими как *Ethernet*, *Wi-Fi* и *Bluetooth*, для подключения к сети и другим устройствам. Это позволяет использовать *Banana Pi* в различных *IoT*-

проектах, системах видеонаблюдения или в качестве медиацентра, который может подключаться к интернету и в потоках воспроизводить контент.

Как и *Raspberry Pi*, *Banana Pi* поддерживает различные операционные системы, включая *Linux* и *Android*.

Banana Pi представляет собой удобный и доступный вариант одноплатного компьютера, который может быть использован для различных проектов, особенно если не требуются высокие вычислительные мощности или если бюджет ограничен.

Для реализации задачи управления движения роботом рекомендуется использовать микрокомпьютер *Asus Tinker Board*. Он отличается от описанных выше аналогов более мощным процессором и графическим ускорителем, что делает его идеальным выбором для проектов, требующих высокой производительности. *Asus Tinker Board* также обладает большим количеством портов *USB* и поддержкой *Gigabit Ethernet*, что делает его более гибким для подключения внешних устройств и работой в сетях высокой скорости.

1.4 Сравнительная характеристика *GPS*-модулей для нахождения положения робота

Для определения позиции робота необходим *GPS*-модуль. Далее будут рассмотрены наиболее популярные *GPS* -модули:

– *GPS*-модуль *u-blox NEO*-7*M*: это *GPS*-модуль с высокой точностью и высокой скоростью обновления данных. Он также имеет низкий уровень шума и потребление энергии. Однако, он может быть более дорогим, чем *GPS Control Neo*-6*M.*

– *GPS*-модуль *VK*2828*U*7*G*5*LF*: это *GPS*-модуль с низкой стоимостью и высокой точностью. Он также имеет компактный размер и легко подключается к микроконтроллеру. Однако, он может иметь более низкую скорость обновления, чем *GPS Control Neo*-6*M*.

– *GPS*-модуль *Adafruit Ultimate GPS Breakout*: это *GPS*-модуль с высокой точностью и поддержкой различных протоколов, включая *NMEA* и *UBX*. Он также имеет встроенный антенный усилитель для увеличения дальности сигнала. Однако, он может быть более дорогим, чем *GPS Control Neo*-6*M*.

– *GPS*-модуль *Beitian BN*-180: это *GPS*-модуль с низкой стоимостью и высокой точностью. Он также имеет компактный размер и низкое потребление энергии. Однако, он может иметь более низкую скорость обновления, чем *GPS Control Neo*-6*M*.

В целом, выбор *GPS*-модуля будет зависеть от конкретных потребностей и требований приложения. *GPS Control Neo*-6*M* является хорошим выбором, когда

необходима высокая точность, высокая скорость обновления и низкая стоимость, но есть и другие аналоги, которые могут подходить для других приложений.

По сравнению с другими *GPS*-модулями на, *GPS Control Neo*-6*M* имеет ряд преимуществ:

– использует 50 каналов для получения данных о спутниках *GPS*, что позволяет достичь высокой точности в определении местоположения;

– обновляет данные о местоположении и времени каждую секунду, что позволяет быстро реагировать на изменения условий окружающей среды;

– имеет простой интерфейс и может работать с любым микроконтроллером, что делает его очень простым в использовании;

– имеет низкую стоимость по сравнению с другими *GPS*-модулями на рынке, что делает его доступным для широкого круга потребителей.

1.5 Анализ современных средств по формированию маршрутов на спутниковых картах

Существует множество средств по формированию маршрута на спутниковых картах, каждое из которых имеет свои преимущества и недостатки в зависимости от конкретных потребностей пользователя. Ниже приведен краткий обзор и анализ некоторых из наиболее распространенных средств по формированию маршрута на спутниковых картах.

Google Maps является одним из самых популярных инструментов для формирования маршрута на картах. Он имеет обширную базу данных изображений спутниковых снимков, а также предоставляет широкий спектр функций, таких как поиск местоположений, прокладка маршрутов и возможность создания пользовательских карт [2]. Кроме того, *Google Maps* предоставляет *API*, что позволяет разработчикам интегрировать карты в свои приложения.

Bing Maps, разработанный *Microsoft*, является конкурентом *Google Maps* и имеет схожий набор функций. Он также предоставляет доступ к спутниковым изображениям, а также к *3D*-изображениям городов. Однако *Bing Maps* не так широко используется, как *Google Maps*.

ArcGIS, разработанный *Esri*, является мощной ГИС-платформой, которая позволяет пользователям создавать и анализировать спутниковые карты с использованием многих источников данных. *ArcGIS* имеет широкий спектр функций, включая анализ пространственных данных, интеграцию с другими приложениями и создание пользовательских маршрутов на картах.

OpenStreetMap является проектом, который создает свободную, редактируемую картографическую базу данных мира. *OpenStreetMap* предоставляет доступ к спутниковым изображениям, а также к другим слоям информации, таким как данные дорожной сети, населенных пунктов и границ административных

единиц. *OpenStreetMap* используется в различных приложениях и сервисах, в том числе веб-картах и навигационных приложениях.

Mapbox предоставляет платформу для создания и развертывания карт. Он имеет широкий спектр функций, включая доступ к спутниковым изображениям, возможность создания пользовательских маршрутов на картах и интеграцию с другими приложениями. *Mapbox* также предоставляет *API* для интеграции карт в приложения.

Yandex Maps – это набор программных интерфейсов, предоставляемый компанией Яндекс для создания интерактивных карт и геолокации веб-приложений [3]. Он позволяет интегрировать интерактивные карты Яндекса в веб-приложения и сайты, а также добавлять кастомизированные элементы управления, отметки и метки, показывать маршруты и многое другое.

Выбор средства для формирования спутниковых карт зависит от конкретных потребностей пользователя и его бюджета. Например, *Google Maps, Bing Maps и Yandex Maps* являются наиболее подходящими инструментами для небольших бизнесов или отдельных пользователей, которые ищут интуитивно понятный и доступный интерфейс для создания простых маршрутов на картах.

OpenStreetMap и *Mapbox* являются хорошими вариантами для пользователей, которые хотят создать кастомизированные карты или добавить дополнительные данные в карты. *OpenStreetMap* также является полезным для тех, кто хочет использовать данные карт в своих проектах, так как он предоставляет открытый доступ к данным картографии.

Для разработки графических средств по определению маршрута движения робота будут использоваться *Yandex Maps* так как они имеют ряд преимуществ.

Yandex Maps имеет бесплатный и платный тарифы. Бесплатный тариф ограничен по количеству запросов в день и используется в основном для небольших проектов и демонстрационных целей. Платный тариф позволяет получить больше возможностей, включая большее количество запросов, расширенную функциональность и поддержку.

Yandex Maps обеспечивает высокую точность геокодирования, широкий набор географических данных и возможность создавать многомасштабные карты. Он также предоставляет обширную документацию и примеры кода для различных языков программирования, что упрощает интеграцию карт в приложения.

Yandex Maps – это мощный инструмент для создания интерактивных маршрутов на картах и геолокации веб-приложений, который может быть полезным для различных проектов, особенно для тех, которые требуют высокой точности геокодирования и широкого набора географических данных, что необходима для разработки программного обеспечения по определению маршрута для колесного робота.

2 АРХИТЕКТУРА СИСТЕМЫ ПО ОБЕСПЕЧЕНИЮ ДВИЖЕНИЯ АГРАРНОГО РОБОТА ПО ЗАДАННОЙ ТРАЕКТОРИИ

2.1 Архитектура компонентов аппаратной системы

Архитектура компонентов аппаратной системы представлена на рисунке 2.1.

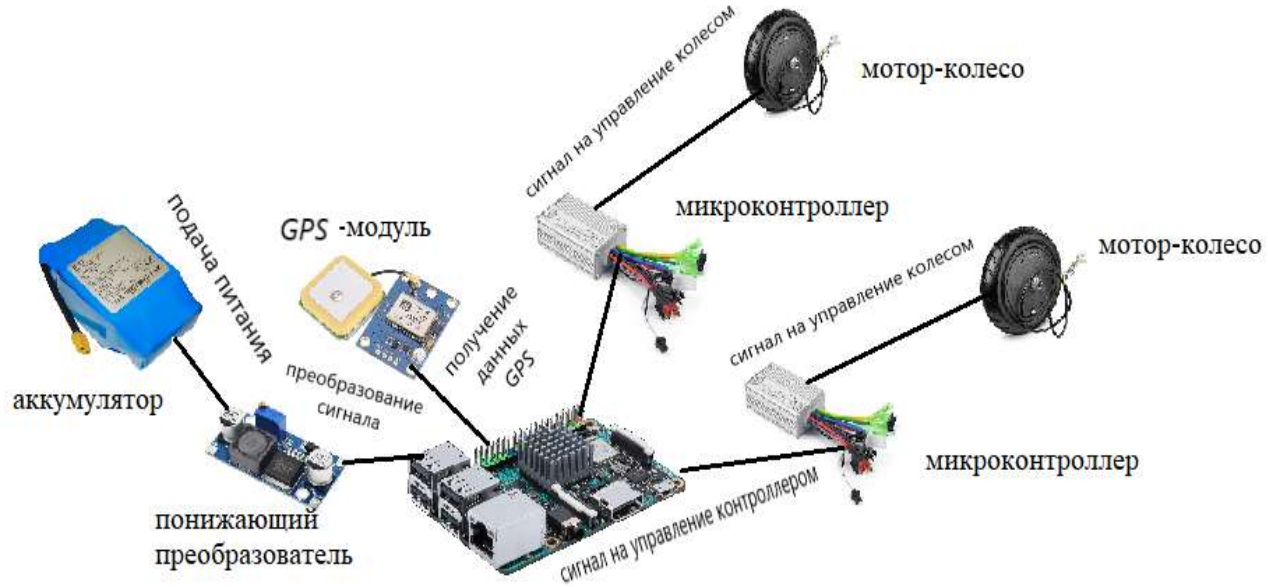

Рисунок 2.1 Архитектура компонентов аппаратной системы

Структура компонентов аппаратной системы состоит из:

– одноплатного компьютера;
– *GPS*-модуля;
– аккумулятора;
– понижающего преобразователя;
– двух микроконтроллеров;
– двух мотор-колес.

У каждого компонента системы свои задачи и цели.

Универсальные контроллеры применяются в системах управления электрическими мотор-колесами. Они обеспечивают ряд функций, связанных с управлением мощностью и скоростью мотор-колеса, а также гарантируют защиту и контроль системы.

Универсальные контроллеры для мотор-колес осуществляют управление мощностью, контролируя ток и напряжение, поступающие в мотор. Это позво-

ляет регулировать скорость мотора и обеспечивать оптимальную производительность.

Они также обеспечивают возможность регулировки скорости мотора в соответствии с требованиями пользователя. Путем контроля частоты импульсов, подаваемых на мотор, универсальные контроллеры позволяют изменять скорость и обеспечивать плавное ускорение и замедление.

Универсальные контроллеры считывают обратную связь от мотора, например, информацию о скорости и положении ротора. Они используют эту информацию для обеспечения точного контроля мотора и поддержания заданной скорости или положения.

Важной функцией микроконтроллеров является защита от перегрузок и ошибок, связанных с работой мотора. Они могут обнаруживать перегрузки, а также контролировать ток, напряжение и температуру, чтобы предотвратить повреждение компонентов и перегрев системы.

Универсальные контроллеры обеспечивают возможность коммуникации с другими устройствами или системами через различные интерфейсы, такие как *UART*, *SPI* или *I2C*. Это позволяет интегрировать мотор-колеса с другими устройствами и реализовывать дополнительные функции, например, системы управления или мониторинга.

Они являются программируемыми, что позволяет настраивать их поведение под конкретные требования.

Они могут быть запрограммированы для реализации различных алгоритмов управления и функций без необходимости в физической перенастройке или замене компонентов.

Оптимизация энергопотребления является важным аспектом микроконтроллеров для мотор-колес.

GPS-модуль, в связке с одноплатным компьютером, предоставляет ряд функций и особенностей, которые позволяют использовать глобальную систему позиционирования в различных приложениях и проектах.

Этот модуль способен определить текущее местоположение на основе данных, полученных от спутников *GPS*, что позволяет отслеживать перемещение объектов или точно определить положение пользователя.

Он также обеспечивает навигационные возможности, позволяя определить маршрут до заданной цели и предоставлять информацию о направлении и расстоянии до нее.

GPS-модуль также предоставляет точное время синхронизации на основе атомных часов спутников *GPS*, что может быть полезно для синхронизации систем, событий и устройств, требующих точного времени. Он добавлять географическую информацию к фотографиям, видео и другим данным, что упрощает их организацию и отслеживание на основе местоположения.

GPS-модуль можно использовать для мониторинга и отслеживания движения объектов или людей.

В целом, *GPS*-модуль с одноплатным компьютером предоставляет разнообразные функции, которые можно использовать в зависимости от конкретных потребностей проекта или приложения, и способствует развитию различных областей, связанных с позиционированием и навигацией.

Понижающий преобразователь является электронным устройством, которое используется для преобразования постоянного напряжения с более высокого уровня до более низкого уровня. Он выполняет эту функцию путем регулирования и управления током в цепи, чтобы создать желаемый выходной уровень напряжения.

Основная функция понижающего преобразователя состоит в том, чтобы принимать постоянное напряжение, которое поступает на вход, и уменьшать его до требуемого уровня на выходе. Это может быть полезно, когда необходимо подать питание на устройство или систему, которая работает на более низком напряжении, чем источник питания. Например, если есть источник питания с напряжением 12 В, а устройство требует питания 5 В, понижающий преобразователь может быть использован для снижения напряжения с 12 В до 5 В, чтобы обеспечить соответствующее питание.

Понижающие преобразователи обеспечивают высокую эффективность, что означает, что они минимизируют потери энергии в процессе преобразования напряжения. Более высокая эффективность ведет к меньшему количеству отпущенной тепловой энергии и позволяет экономить электрическую энергию.

Понижающие преобразователи обычно обеспечивают стабильный выходной ток и напряжение, что важно для надежной работы устройств и систем. Они могут быть спроектированы для работы с различными входными и выходными напряжениями, что позволяет им быть универсальными и применимыми во многих различных ситуациях.

Преобразователи обладают функциями, такими как защита от перегрузки, короткого замыкания и перегрева. Это помогает предотвратить повреждение устройств и систем, а также обеспечивает безопасность при работе с устройством.

Мотор-колёса для роботов обеспечивают привод движения и управление скоростью. Они позволяют роботу маневрировать и изменять направление движения. Колеса также могут использоваться для регулировки положения робота и обладают сенсорной обратной связью для более точного контроля движения.

Некоторые колеса имеют функцию регенеративного торможения, преобразуя кинетическую энергию в электрическую. Они интегрируются в систему управления роботом и обеспечивают экономию пространства благодаря своей

компактности. Главная функция мотор-колес для роботов заключается в обеспечении движения, маневренности и контроля положения и скорости робота.

Одноплатный компьютер служит центральным устройством управления и координации в системе. Он выполняет функцию обработки данных и принятия решений на основе полученной информации от универсальных контроллеров, мотор-колес, аккумулятора и датчика *GPS*. Это позволяет создать единую интеллектуальную систему, способную эффективно управлять и координировать все компоненты.

Одноплатный компьютер считывает данные от универсальных контроллеров, которые управляют мотор-колесами. Он анализирует информацию о скорости, токе, напряжении и других параметрах, необходимых для эффективного управления двигателями. На основе этих данных компьютер принимает решения о регулировании скорости, управлении моментом и других параметрах работы мотор-колес.

Кроме того, одноплатный компьютер может контролировать состояние и управлять АКБ. Он считывает информацию о заряде и напряжении АКБ, что позволяет определить оставшуюся емкость и прогнозировать остаточное время работы.

Компьютер также может управлять процессом зарядки АКБ, контролировать токи и напряжение, чтобы обеспечить оптимальные условия зарядки и продлить срок службы батарей.

Датчик *GPS* предоставляет информацию о местоположении и скорости движения. Одноплатный компьютер использует эти данные для определения текущего положения транспортного средства и планирования оптимального маршрута.

Компьютер может анализировать данные о скорости и координатах для определения движения и принятия решений об оптимальном использовании энергии.

Одноплатный компьютер обеспечивает связь между всеми компонентами системы. Он передает команды и данные от контроллеров к мотор-колесам, контролирует взаимодействие между устройствами и обеспечивает передачу информации о состоянии и настройках.

Кроме того, компьютер может обрабатывать данные с нескольких источников одновременно и координировать их взаимодействие, чтобы обеспечить согласованную работу всей системы.

Одноплатный компьютер играет роль интеллектуального центра управления и координации в системе с двумя универсальными контроллерами для мотор-колес, аккумулятором, двумя мотор-колесами и датчиком *GPS*. Он обрабатывает данные, принимает решения и обеспечивает оптимальное функционирование всей системы, повышая ее эффективность, безопасность и надежность.

2.2 Аппаратное обеспечение одноплатного компьютера

В качестве операционной системы на одноплатном компьютере используется *Debian* 10.

Debian 10, также известный как *Debian GNU*/*Linux* 10 или просто *Debian Buster*, является одним из самых популярных дистрибутивов операционной системы *Linux*. *Debian* – это свободная и открытая операционная система, основанная на ядре *Linux*, и предлагает широкий выбор пакетов программного обеспечения, разработанных сообществом разработчиков.

Debian 10 был выпущен в июле 2019 года и в настоящее время является стабильной версией *Debian*. Он предлагает множество новых функций, улучшений производительности и обновленных пакетов программного обеспечения.

Одним из ключевых преимуществ *Debian* 10 является его высокая стабильность и надежность. Он был разработан с фокусом на обеспечение стабильной работы системы, что делает его идеальным выбором для серверов и других критически важных систем. Надежность *Debian* 10 минимизирует возможные сбои и проблемы.

Операционная система *Debian* 10 поставляется с обширным набором программного обеспечения. В его репозиториях находится широкий выбор пакетов, включая веб-серверы, базы данных, графические инструменты, офисные приложения и многое другое. Богатый выбор программного обеспечения позволяет настроить систему под свои потребности, предлагая гибкость и разнообразие инструментов.

Debian обеспечивает активную поддержку сообщества разработчиков, что обеспечивает регулярные обновления безопасности. *Debian* 10 получает постоянные обновления, которые исправляют уязвимости и проблемы безопасности. Это важно для обеспечения защиты системы и данных от внешних атак и угроз.

Debian имеет обширную документацию и активное сообщество пользователей.

Debian 10 представляет собой надежную, гибкую и безопасную операционную систему с широким набором программного обеспечения и удобным процессом установки. Он отлично подходит для серверов и других критически важных систем, а также для широкого спектра пользователей с разным уровнем опыта.

Для удаленного доступа к одноплатному компьютеру может использоваться *VNC*.

VNC – это графический протокол, который позволяет удаленному пользователю управлять и просматривать графический интерфейс удаленной системы. Он используется для удаленного доступа к рабочему столу компьютера или сервера.

VNC работает на клиент-серверной модели, где сервер запускается на удаленной машине, а клиентское приложение используется для подключения к серверу и отображения его графического интерфейса.

Одной из главных особенностей *VNC* является его способность работать с различными операционными системами. Это означает, что пользователи могут подключаться к компьютерам или серверам с операционными системами, такими как *Linux*, *Windows*, *macOS* и *Debian* 10 в том числе, и управлять ими из любого устройства с поддержкой *VNC*.

Одной из важных аспектов *VNC* является безопасность передачи данных. Некоторые варианты *VNC* поддерживают шифрование данных, что обеспечивает конфиденциальность и защиту при передаче информации через сеть. Это важно для предотвращения несанкционированного доступа к удаленному компьютеру или серверу.

VNC находит применение в различных областях. Администраторы систем используют его для управления удаленными компьютерами и серверами, что упрощает обслуживание и настройку систем. Техническая поддержка может использовать *VNC* для удаленного доступа к компьютерам пользователей и решения их проблем. *VNC* также может быть использован в образовательных целях для удаленного обучения и демонстрации процессов на экране.

VNC является мощным инструментом для удаленного доступа и управления компьютерами и серверами. Он предоставляет гибкость и удобство удаленной работы, а также обеспечивает защиту данных при передаче через сеть.

VNC в связке с *Debian* 10 обеспечивает возможность удаленного управления одноплатным компьютером без необходимости его подключения к монитору.

Путем использования *VNC* могут быть изменены или добавлены скрипты управления одноплатным компьютером, обновлено программное обеспечение и выполнены другие действия из удаленной системы.

2.3 Архитектура программного обеспечения по формированию маршрута

Для реализации ПО выбрана клиент-серверная архитектура с использованием *TCP*-сокетов в качестве механизма сетевого взаимодействия. Этот выбор обусловлен простотой и эффективностью данной модели, а также тем фактом, что клиент играет роль инициатора взаимодействия, в то время как сервер отвечает на запросы клиента.

Клиент и сервер обмениваются сообщениями по принципу запрос-ответ через *TCP*-сокеты. Клиент отправляет запрос на сервер, указывая адрес и порт,

по которому сервер доступен. Сервер принимает запрос, обрабатывает его и формирует ответ, который отправляется обратно клиенту через тот же *TCP*-сокет.

Для облегчения взаимодействия между клиентом и сервером и упрощения межплатформенного обмена данными сервер реализует *API* – интерфейс прикладного программирования.

API предоставляет уровень абстракции, который позволяет клиенту обращаться к сервису, не беспокоясь о деталях его реализации. Он также обеспечивает определенный формат контента для обмена данными, что способствует согласованности и пониманию между клиентом и сервером.

Схема взаимодействия системы по формированию маршрута и системы по обеспечению движения робота показана ни рисунке 2.2.

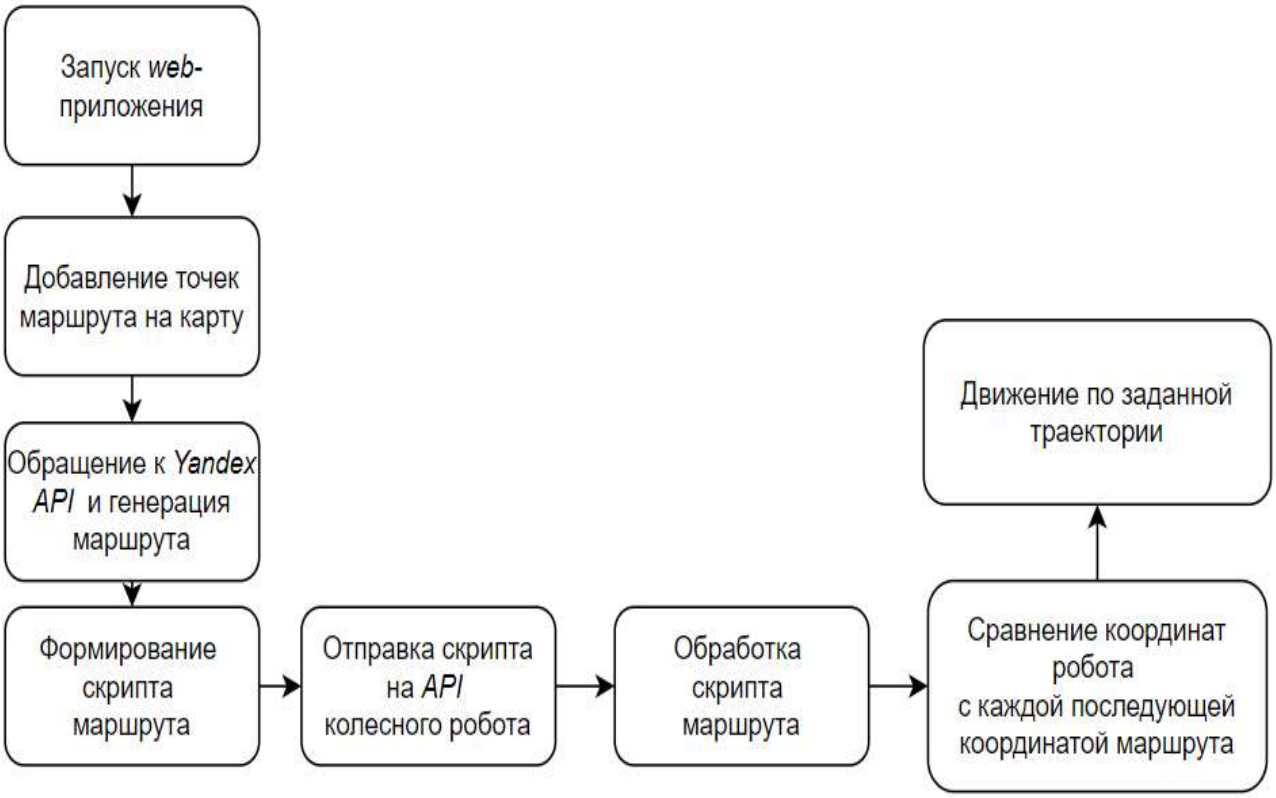

Рисунок 2.2 Схема взаимодействия системы по формированию маршрута и системы по обеспечению движения робота

Оптимизация взаимодействия между клиентом и сервером достигается благодаря использованию постоянного *TCP*-соединения.

После установки соединения между клиентом и сервером, оно остается активным и используется для последующих запросов и ответов.

Это помогает избежать накладных расходов на установку и разрыв соединения для каждого запроса, что способствует более эффективному обмену данными.

Архитектура программной системы по обеспечению движения колесного робота по заданной траектории представлена на рисунке 2.3.

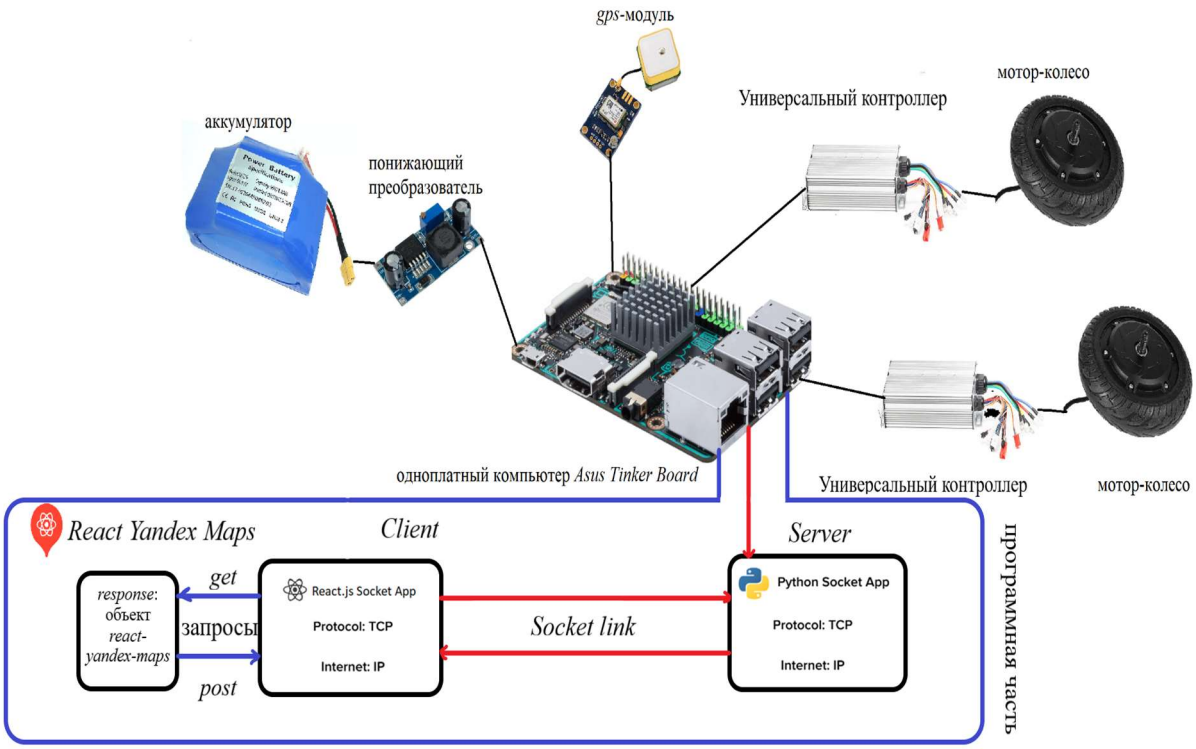

Рисунок 2.3 – Архитектура программной системы по обеспечению движения колесного робота по заданной траектории

Таким образом, клиент-серверная архитектура с использованием *TCP*-сокетов и *API* обеспечивает надежное и эффективное сетевое взаимодействие между клиентом и сервером. Клиент отправляет запросы на сервер, сервер их обрабатывает и формирует ответы, которые возвращаются клиенту. Этот подход обеспечивает стабильность, простоту и удобство взаимодействия между приложениями, использующими данную архитектуру.

Клиентское взаимодействие с *Yandex API* осуществляется при помощи *http*-запросов. Клиент использует *get*-запрос с параметром, для получения объекта *react-yandex-maps.* В качестве параметра выступает ключ (*API key*), получаемый после регистрации в кабинете разработчика *Yandex.* После обращения к *Yandex API,* клиент получает в качестве ответа объект *react-yandex-maps,* содержащий набор методов для работы с картами.

Клиентское взаимодействие с сервером через сокеты представляется следующим образом:

– клиент создает сокет и устанавливает соединение с сервером, используя сетевой адрес сервера и порт, к которому он привязан;

– клиент отправляет запрос на сервер, используя установленное соединение и форматируя запрос в соответствии с документацией *API*;

– сервер принимает запрос и проводит аутентификацию клиента для проверки прав на выполнение запроса;

– сервер обрабатывает запрос и генерирует ответ, включая информацию о том, был ли запрос успешным, а также запрошенные клиентом данные;

– сервер отправляет ответ клиенту, используя тот же сокет, через который был получен запрос;

– когда взаимодействие между клиентом и сервером завершено, клиент закрывает соединение, и сокет освобождается для дальнейшего использования.

Преимущества, которые предлагает данная архитектура:

– системы, реализующие *REST API*, могут эффективно масштабироваться благодаря оптимизации взаимодействия между сервером и клиентом по *REST*, отсутствие сохранения состояния снимает нагрузку с сервера: серверу не нужно сохранять информацию о предыдущих запросах клиента;

– веб-службы *REST* поддерживают полное разделение клиента и сервера, упрощают и разделяют различные серверные компоненты, чтобы каждая часть могла развиваться независимо, изменения платформы или технологии в серверном приложении не влияют на клиентское приложение;

– *REST API* не зависит от используемой технологии, можно создавать как клиентские, так и серверные приложения на разных языках программирования, не затрагивая структуру *API*. Также можно изменить базовую технологию на любой стороне, не влияя на обмен данными.

2.4 Структура программных средств для обеспечения движения колесного робота

Структура программных средств для обеспечения движения колесного робота включает несколько компонентов, которые работают вместе для запуска движения и управления колесами.

На самом низком уровне находятся двигатели, которые получают цифровой аналоговый. Чтобы управлять скоростью и направлением вращения колес, сигнал проходит через процесс изменения с помощью широтно-импульсной модуляции. ШИМ представляет собой метод модуляции, при котором длительность импульсов сигнала изменяется для управления выходной мощностью двигателей. Это позволяет точно регулировать скорость и направление колес.

Затем аналоговый сигнал после ШИМ передается на контроллеры, которые отвечают за запуск колес робота. Контроллеры выполняют логику управления

движением, определяющую параметры движения, такие как скорость, ускорение, торможение и повороты. Они работают на основе предварительно заданных алгоритмов и получают команды от вычислительного устройства – одноплатного компьютера.

Между контроллерами и одноплатным компьютером установлены фильтры низких частот.

Схема фильтра низких частот показана на рисунке 2.4.

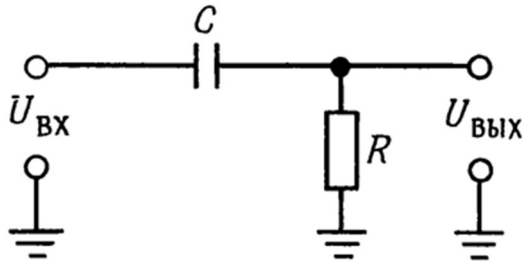

Рисунок 2.4 – Схема фильтра низких частот

Фильтры низких частот представляют собой электронные устройства, которое позволяют проходить сигналам с низкими частотами и ограничивают прохождение высокочастотных сигналов [4, с. 38]. В данном случае фильтры служат для удаления или снижения шумов и помех, которые могут возникать в электрической цепи между контроллерами и одноплатным компьютером. Это помогает обеспечить более стабильную и точную передачу команд управления между компонентами системы.

Для начала, двигатели робота подключены к *GPIO* портам одноплатного компьютера. *GPIO* порты предоставляют цифровые сигналы, которые быть использованы для управления внешними устройствами, такими как двигатели. Обычно *GPIO* порты представляют собой пины, которые могут быть настроены как входные или выходные, в зависимости от требуемой функциональности.

Для передачи сигнала на двигатели через ШИМ, на одноплатный компьютер используются поддерживаемые *GPIO* порты, которые имеют функциональность ШИМ. Настройка *GPIO* портов в режиме ШИМ позволяет генерировать сигналы с изменяемой длительностью импульсов, которые управляют скоростью и направлением вращения колес.

Одноплатный компьютер выполняет алгоритм управления и отправляет соответствующие сигналы управления на *GPIO* порты, связанные с двигателями. Это позволяет реализовать логику управления движением колес на уровне программного обеспечения.

Таким образом, структура программного обеспечения для обеспечения движения колесного робота включает двигатели, работающие на основе ШИМ, контроллеры, отвечающие за логику управления движением колес, а также фильтрами низких частот, которые обеспечивает стабильную связь между контроллерами и одноплатным компьютером.

2.5 Схема подключения компонентов к одноплатному компьютеру

На рисунке 2.5 приведена схема подключения компонентов аппаратной системы.

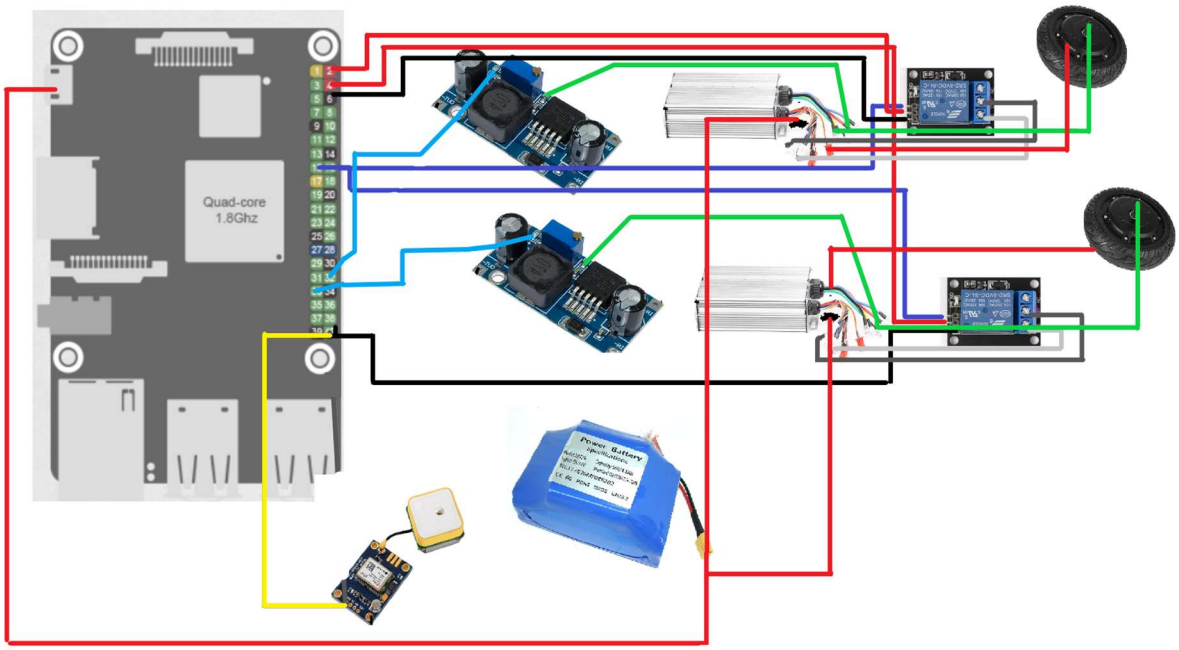

Рисунок 2.5 – Схема подключения компонентов робота

Аккумулятор подает питание на одноплатный компьютер и на универсальные контроллеры, через которые питание попадает на мотор-колеса.

Между универсальными контроллерами и мотор-колесами установлено реле, позволяющее изменять направление мотор-колес, путем получения сигнала на реле и замыкания контакта.

Одноплатный компьютер управляет контроллерами через понижающие преобразователи трансформируя цифровой сигнал в аналоговый, после чего контроллеры подают сигнал на управление колесами.

Понижающие преобразователи подключаются к одноплатному компьютеру по специальным пинам, поддерживающим ШИМ.

Датчик *GPS* подключается в любой свободный пин, не отвечающий за питание.

Описание цветовых линий к рисунку 2.5:

– красные линии обозначают подключение питания (плюс);

– черные линии – земля;

– фиолетовые линии обозначают подключение реле к одноплатному компьютеру по свободным пинам, не отвечающим за питание;

– голубые линии отвечают за подключение понижающих преобразователей к одноплатному компьютеру и обозначают землю;

– желтая линия – подключение *GPS*-датчика к одноплатному компьютеру;

– зеленые линии отвечают за подключение понижающих преобразователей к мотор-колесам, через универсальные контроллеры;

– серые линии и темно-серые линии – соединяют реле с контактами на универсальных контроллерах, которые отвечаю за смену направления вращения мотор-колес.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ ОБЕСПЕЧЕНИЯ ДВИЖЕНИЯ АГРАРНОГО РОБОТА ПО ЗАДАННОЙ ТРАЕКТОРИИ

3.1 Клиентская часть программного обеспечения движения колесного робота по заданной траектории

Клиентская часть программного обеспечение по формированию маршрута на спутниковой карте для колесного робота разрабатывалась при помощи следующих инструментов:

– *ReactJs* – отвечает за графическую составляющую программного комплекса.

– *Yandex maps API* – отвечает за инструменты позволяющие работать с картой.

ReactJs предлагает простое и функциональное создание компонентов, а также пропагандирует их использование для поддержания элегантного кода *API* [5, с. 20].

Имея большой выбор доступных открытых плагинов и расширений, можно разработать практически любой тип веб-сайта.

Особенности *ReactJs*:

– компонентно-ориентированный;

– декларативный;

– производительный (благодаря *React Virtual DOM*);

– серверный рендеринг.

Схема *React* компонентов изображена на рисунке 3.1.

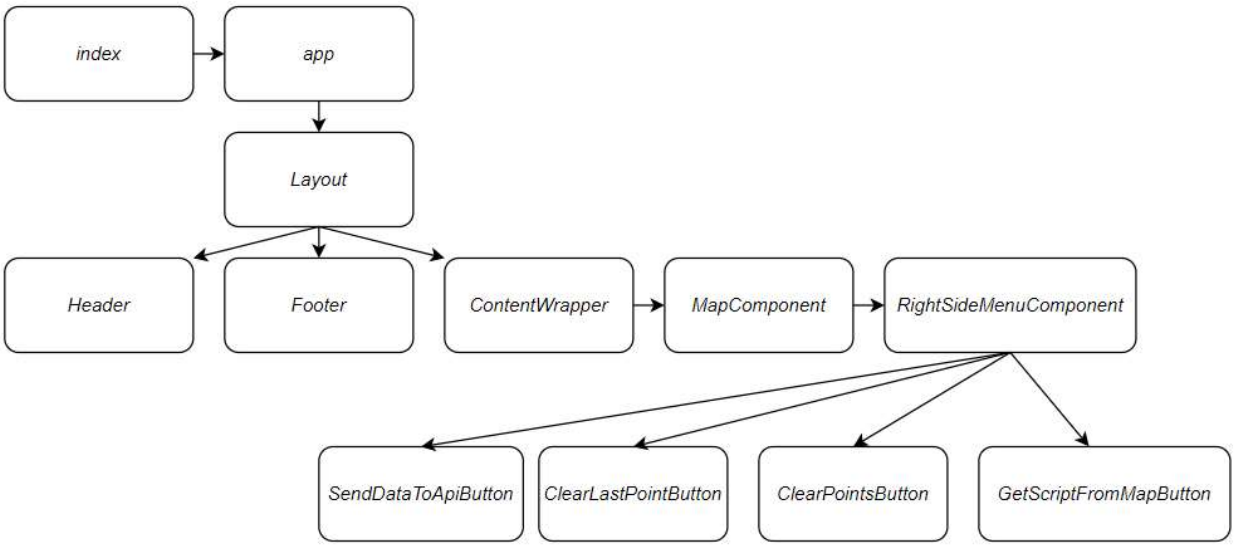

Рисунок 3.1 – Схема *React* компонентов

В целом *ReactJs* обладает хорошей производительностью, некорректное использование или неправильное проектирование компонентов может привести к ухудшению производительности приложения. Например, неправильная обработка обновлений состояния или неправильная оптимизация компонентов может привести к лишним перерисовкам или увеличению нагрузки на виртуальный *DOM*.

Программное обеспечение представляет собой клиентское приложение предоставляя следующие возможности:

– добавлять ограниченное количество точек маршрута на карте (количество точек можно задать, но не больше того числа, которое предоставляет *Yandex maps API*);

– переключать вид карты – спутниковая гибрид или схематичный;

– очищать все точки маршрута на карте;

– удалить последнюю добавленную точку маршрута;

– сгенерировать скрипт маршрута в формате *JSON* с возможностью экспорта. Скрипт маршрута хранит долготу, широту и номер каждой заданной координаты, а также начальное положение карты и кратность зума.

– Возможность отправки скрипта на *API*, который установлен на серверной части программного обеспечения.

Сгенерированный скрипт маршрута карты представлен на рисунке 3.2.


```
[
    {
        "_id": 1,
        "cords": {
            "lat": 52.406875671585105,
            "lang": 30.93598553291818
        }
    },
    {
        "_id": 2,
        "cords": {
            "lat": 52.40477592738968,
            "lang": 30.937165704884727
        }
    },
    {
        "_id": 3,
        "cords": {
            "lat": 52.404572509334194,
            "lang": 30.935513464131553
        }
    },
        {
        "_id": 4,
        "cords": {
            "lat": 52.4045725099999913,
            "lang": 30.935513464131123
        }
    }
]
```


Рисунок 3.2 – Сгенерированный скрипт карты в формате *JSON*

Для добавления элементов управления на саму карту использовались специальные объекты:

– *YMaps* – объект, предоставляющий доступ к самой *Yandex API;*

– *Map* – объект самой карты;

– *SearchControl* – элемент управления, для поиска местоположения объекта;

– *GeolocationControl* – элемент управления, для получения текущей геопозиции;

– *TypeSelector* – элемент управления, для выбора типа карты.

Для доступа к *Yandex API* необходимо зарегистрироваться и войти в аккаунт разработчика на официальном сайте *Yandex*. Далее подключить бесплатный тариф и получить ключ доступа – *API KEY*. Этот ключ доступа вставляется в *url*-подключения к *Yandex maps API* объекта *YMaps.* Так же в этот объект добавляются параметры языка, например – *RU-ru.*

Функциональная схема *React*-приложения показана на рисунке 3.3.

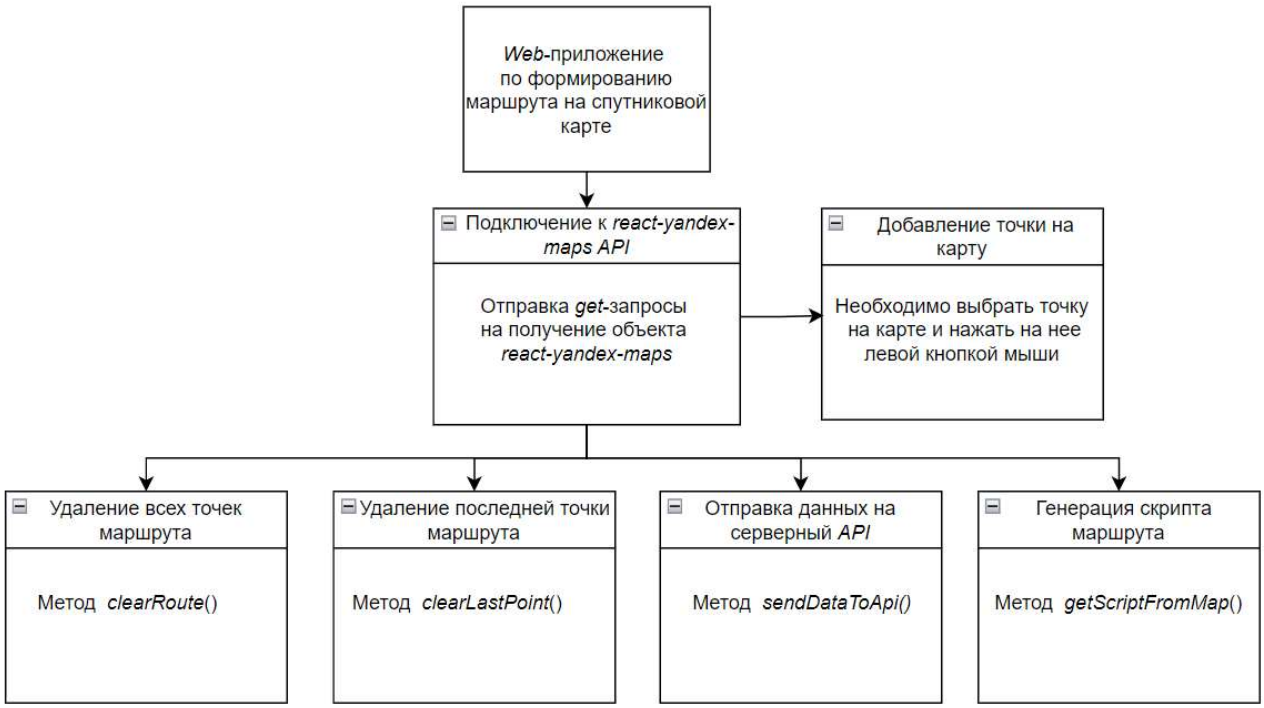

Рисунок 3.3 – Функциональная схема *React*-приложения

Для инициализации карты необходимо задать параметры:

– *RouteMode* – тип построения маршрута. Он должен быть одним из следующих типов:

1) «*pedestrian*» – пешеходная маршрутизация. Доступна только для мультимаршрутов (опция *multiRoute* должна быть выставлена в *true*);

2) «*masstransit*» – маршрутизация с использованием общественного транспорта. Доступна только для мультимаршрутов (опция *multiRoute* должна быть выставлена в *true*);
3) «*auto*» – автомобильная маршрутизация.

– *MapState* – начальное положение на карте. *MapState* включает в себя следующие параметры: объект *center,* который содержит в себе информацию о геопозиции в виде данных долготы и широты; объект *zoom* – отвечает за кратность зума на карте.

Для хранения состояния карты и маркеров на ней использовались *React*-функции.

В клиентском приложении использовались следующие функции:

– *useState* – это функция в *React*, которая позволяет добавлять локальное состояние в функциональные компоненты. Он используется для объявления переменной состояния и функции для ее обновления. Функция возвращает массив, состоящий из текущего значения состояния и функции для его обновления. При вызове функции *setState* с новым значением, *React* обновит компонент и вызовет перерисовку.

– *useRef* – это функция в *React*, которая позволяет создавать изменяемые переменные, которые могут быть использованы в компонентах. Он используется для получения доступа к *DOM*-элементам и для сохранения любой другой мутабельной переменной между рендерами компонента. Функция возвращает объект со свойством *current*, которое содержит текущее значение переменной. Значение можно изменять напрямую, обновляя свойство *current*.

Для взаимодействия с *API* робота, при наличии интернет соединения, использовались сокеты – это механизм для обмена данными между сервером и клиентом в режиме реального времени.

В *React* можно использовать сокеты для обмена данными с сервером или другими клиентами. Для этого необходимо установить библиотеку *Socket.io*, которая предоставляет простой и удобный интерфейс для работы с сокетами.

Далее будут рассмотрены компоненты *React*-приложения клиентской части программного обеспечения.

ButtonsComponets – компоненты отвечающие за взаимодействие с кнопками графического интерфейса включают:

– *ClearLastPointButton* – очищает последнюю добавленную точку на карту маршрута;

– *ClearPointsButton* – очищает все точки на карте маршрута;

– *GetScriptFromMapButton* – формирует скрипт, содержащий набор координат точек, отмеченных на карте маршрута;

– *SendDataToApiButton* – отвечает за отправку координат на сервер, установленный на колесном роботе.

Компонент *RightSideMenu* – отвечает за конфигурацию и расположение меню в графическом интерфейсе.

Компонент *Constants* содержит в себе константы:

– *API_KEY* – ключ для доступа к функциям *yandex-react-api*;

– *ROUTE_MODE* – настройка типа маршрута на карте;

– *CURRENT_MAP_STATE* – текущее состояние карты;

– *MAX_MARKER_NUMBERS* – максимальное количество маркеров на карте;

– *API_URI* – ссылка ресурс *API Yandex maps.*

Так же был написан сервис для взаимодействия с серверной частью программного обеспечения движения колесного робота по заданной траектории – *sendDataToApi.*

Объявлена переменная *socket*, устанавливает соединение с сервером, используя *URI API_URI*.

Внутри функции *sendDataToApi* определена функция *sendData*, которая проверяет, является ли *data* массивом и имеет ли он длину больше нуля. Если условие выполняется, то происходит отправка данных на сервер через вызов *socket.emit*, где данные преобразуются в строку с помощью встроенного метода *JSON.stringify*.

Если условие не выполняется, выводится сообщение об ошибке в консоль.

Объект *eventHandlers* содержит обработчики для различных событий, которые могут возникнуть при работе с сокетом. В данном случае, определены обработчики для событий «*data*», «*response*», «*connect_error*» и «*error*».

При помощи метода *socket.on*, каждому событию из объекта *eventHandlers* присваивается соответствующая функция-обработчик.

Компонент *MapComponent*, использует библиотеку *react-yandex-maps* для отображения карты с возможностью добавления маркеров и построения маршрутов.

В компоненте импортируются необходимые зависимости, такие как *React*, *useState*, *useRef* из *React*, а также несколько компонентов и объектов из *react-yandex-maps*. Импортируются также некоторые константы из файла *constants* и компонент *RightSideMenu*.

В компоненте объявляются следующие переменные и состояния:

– *ymaps* – состояние, хранящее объект карты из библиотеки *react-yandex-maps*;

– *markers* – состояние, хранящее массив маркеров на карте;

– *savedMarkers*: переменная, которая создается путем маппинга массива *markers* и преобразования его в объекты, содержащие *_id* (*id* маркера) и *coords* (координаты маркера);

– также в компоненте используется хук *useRef* для создания переменной *multiRoute*, которая будет использоваться для отслеживания и обновления маршрута на карте.

Далее, определены несколько функций:

– *addPointToMap* – функция, которая добавляет новый маркер на карту при клике на карту, координаты маркера получаются из события клика;

– *getRoute* – функция, которая строит маршрут на карте, используя объект *ymaps* и массив *markers*. Если переменная *multiRoute.current* не существует, создается новый маршрут. Если *multiRoute.current* уже существует, обновляются точки маршрута.

– *buttonHandler* – объект, содержащий методы, которые обрабатывают различные действия, совершаемые пользователем, включая получение скрипта с карты, очистку маршрутов, удаление последней точки и отправку данных на *API*.

Возвращаемый *JSX*-код представляет собой контейнер для карты, обернутый в *YMaps*, содержащий компонент *Map* из *react-yandex-maps*.

Внутри *Map* также используются компоненты:

– *GeolocationControl* – компонент предоставляет пользователю возможность определить свое текущее местоположение на карте. Он отображает кнопку, которую пользователь может нажать, чтобы центрировать карту на своем текущем местоположении.

– *SearchControl* – этот компонент предоставляет пользователю поисковую строку, которую можно использовать для поиска конкретных мест или адресов на карте. Когда пользователь вводит запрос в строку поиска, компонент пытается найти соответствующее место или адрес и отображает его на карте.

– *TypeSelector* – этот компонент предоставляет пользователю возможность выбрать тип карты, например, обычную карту, спутниковое изображение или гибридное представление. Он отображает выпадающий список с доступными типами карты, и пользователь может выбрать нужный тип.

Рядом с картой располагается компонент *RightSideMenu*, который получает *buttonHandler* в качестве входного параметра.

3.2 Серверная часть программного обеспечения движения колесного робота по заданной траектории

Серверная часть программного обеспечения движения колесного робота по заданной траектории состоит из *Python* модулей. А именно:

– *main,* входной модуль, производит начальную настройку и запуск модулей *robotContoller* и *pythonApi* для начала работы системы по обеспечению движения колесного робота по заданной траектории;

– *robotContoller*, модуль логики приложения, отвечает за движение колесного робота;

– *pythonApi*, модуль, обеспечивающий обмен данных с клиентской частью программного обеспечения через сокет;

Схема взаимодействия модулей представлена на рисунке 3.4.

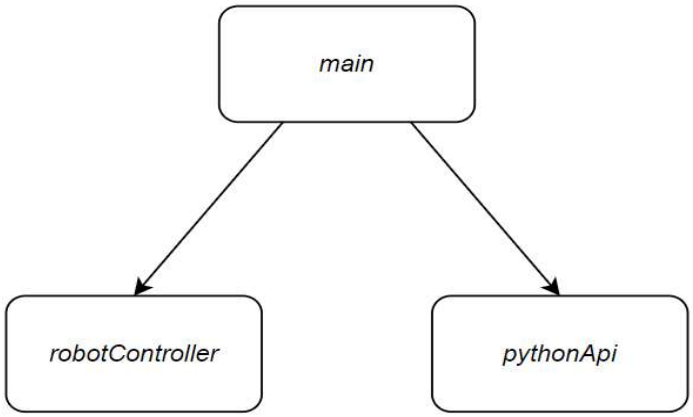

Рисунок 3.4 – Схема взаимодействия модулей

Взаимодействие между клиентским приложением и серверной частью программного обеспечения, установленного на роботе, происходит посредством *TCP*-сокетов.

С клиентского приложения, если присутствует интернет соединение, приходит скрипт маршрута карты. Если интернет соединение отсутствует, то с клиентского приложения экспортируется скрипт, например, на *SD*-карту или флешкарту. Далее этот скрипт переносится на робота и парсится.

3.2.1 Модуль *pythonApi* представлен классами *FileModifiedEventHandler Coord*, а также набором методов для работы с сокетами.

Библиотеки, использующиеся при написании серверной части программного обеспечения *pythonApi* модуля:

– *json* – для работы с *JSON*-данными;

– *os* –для работы с операционной системой (получения пути к файлу, проверки существования файла);

– *Flask* –для создания веб-приложения;

– *flask_cors* – для обеспечения работы с *CORS* – механизмом безопасности, который позволяет веб-странице запрашивать данные с другого источника;

– *flask_socketio* –для реализации двусторонней связи между клиентом и сервером через веб-сокеты;

– *watchdog* – для отслеживания изменений в файловой системе.

API представляет собой серверную часть приложения, написанного на *Python* с использованием *Flask* и *SocketIO*. Это приложение служит для отслеживания изменений в файле, хранящемся на устройстве(роботе), и получения данных с веб-клиента.

Класс *FileModifiedEventHandler* – класс отслеживает изменения в реальном времени в локальном файле, а также метод *on_modified*, который и отслеживает эти изменения.

Класс *FileModifiedEventHandler* наследуется от класса *FileSystemEventHandler* для доступа к функциям, позволяющим взаимодействовать с файловой системой и отслеживать ее состояние в реальном времени [7, с. 56].

Метод *on_modified* вызывает функции *parse_data* или *read_data.*

Метод *read_data* вызывается в том случае, когда файл существует и может быть успешно прочитан.

Метод *parse_data* вызывается в случае, когда файл, находящийся на локальном диске не существует и не может быть прочитан, либо при получении данных с клиента через сокет.

Глобальный метод *watch_file* создает экземпляр класса *FileModifiedEventHandler* и ожидает изменение локального файла.

Класс *Coords* – класс содержащий данный о координатах, полученных с клиента через сокет.

Класс *Coords* имеет следующие поля:

- *id* – порядковый номер координаты;
- *lat* – координата широты;
- *lng* – координата долготы.

Для взаимодействия серверной части приложения с клиентской частью используются сокеты, при наличии выхода в интернет.

Для использования *Socket.IO* в *Python* необходимо установить соответствующую библиотеку, которая обеспечивает поддержку этой технологии в *Python* [6, с. 112]. Одним из таких пакетов является «*python-socketio*».

Класс *Socket.IO* в *Python* предоставляет следующие возможности:

– класс *Socket.IO* позволяет создавать как клиентские, так и серверные сокеты. Для создания клиентского сокета используется метод *socketio.Client()*, а для создания серверного – метод *socketio.Server*();

– метод *connect*() позволяет подключить клиентский сокет к серверу *Socket.IO*. При этом можно указать дополнительные параметры подключения, такие как таймауты и прокси;

– класс *Socket.IO* позволяет определять обработчики событий, которые могут быть отправлены как клиентским, так и серверным сокетом. Для этого используется декоратор *@sio.on('event_name'),* который привязывает функцию-обработчик к определенному событию;

– метод *emit*() позволяет отправлять события от клиентского сокета на сервер *Socket.IO*. При этом можно указать данные, которые необходимо отправить;

– методы *emit*() и *send*() могут использоваться для рассылки событий от сервера *Socket.IO* всем клиентским сокетам, подключенным к нему;

– класс *Socket.IO* предоставляет методы для управления соединением между клиентским и серверным сокетами, такие как *disconnect*(), *connect*() и *reconnect*(). Они позволяют отключать, подключать и переподключаться к серверу *Socket.IO*;

– *Socket.IO* поддерживает протокол веб-сокетов, что обеспечивает более быструю и надежную передачу данных между клиентом и сервером;

– класс *Socket.IO* предоставляет поддержку многопоточности, что позволяет обрабатывать несколько событий одновременно и уменьшает время ожидания ответа от сервера;

– обеспечивает абстракцию над протоколами веб-сокетов, позволяя разработчикам создавать приложения, которые могут работать в широком диапазоне браузеров и устройств.

Информация, обработанная серверным *API,* показана на рисунке 3.5.

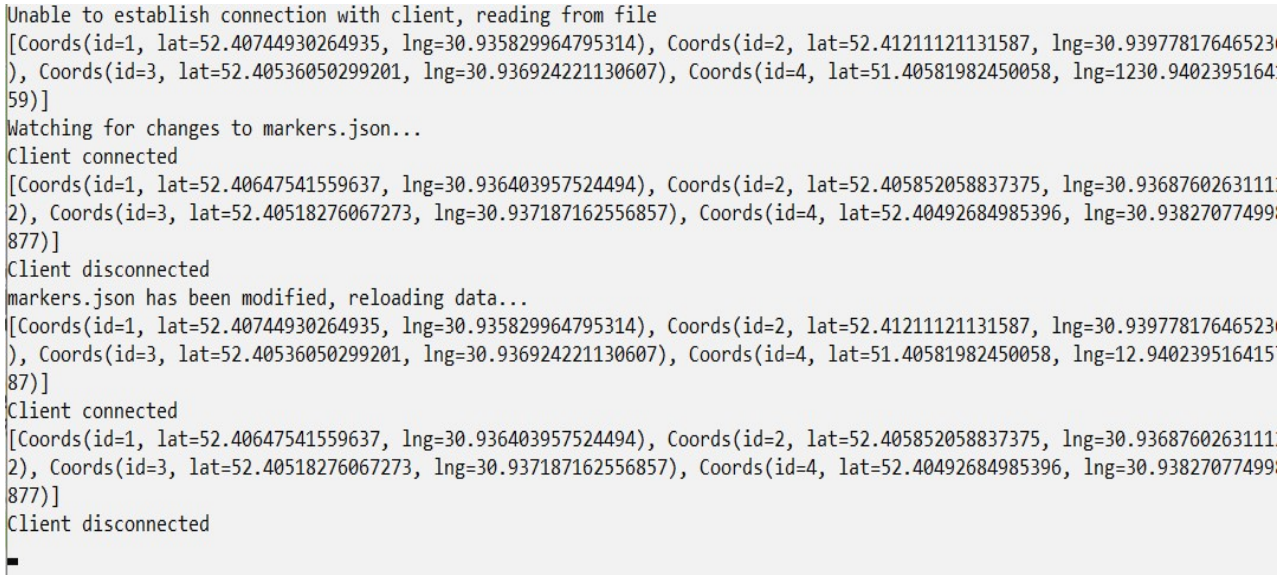

Рисунок 3.5 – Информация обработанная серверным *API*

Для создания сокета в *Socket.IO* необходимо использовать функцию *socketio*.*Client*(), которая создает новый экземпляр клиента *Socket.IO.* При создании экземпляра клиента нужно указать *URL*-адрес сервера, к которому хотите подключиться, а также параметры подключения (например, таймауты, прокси и т.д.).

При запуске сервера, происходит попытка подключения к веб-клиенту с использованием *URI*. Если подключение не удается, происходит чтение данных

из файла и их парсинг. Если данные в скрипте были изменены вручную, то он будет прочитан заново.

Когда на сервер поступает сообщение от клиента, вызывается функция, которая проверяет корректность данных, отправленных клиентом, а затем вызывает функцию для парсинга данных.

Приложение также поддерживает *CORS*, который позволяет другим доменам отправлять запросы на сервер.

Данный сервис отвечает за связь между веб-клиентом и сервером, а также за обновление данных на сервере при изменении файла, хранящего данные о маршруте.

3.2.2 Модуль *robotController* представлен классами *RobotController*

На языке *Python* для работы с *GPIO* на *Asus Tinker Board* используется библиотека *ASUS.GPIO* [8, с.431]. Она предоставляет удобный интерфейс для работы с *GPIO*. Библиотека предоставляет удобный *API* для работы с *GPIO*-пинами.

Библиотеки, использующиеся при написании серверной части программного обеспечения *pythonApi* модуля:

– *ASUS.GPIO* – для взаимодействия с *GPIO*-пинами;

– *GPS* – для работы с геопозицией;

Класс *robotController* содержит следующие поля:

– *pwm_pin*1 – пин управления первым мотор-колесом;

– *pwm_pin*2 – пин управления вторым мотор-колесом;

– *relay_pin*1 – пин управления реле номер один;

– *relay_pin*2 – пин управления реле номер два;

– *relay_pin_value*1 – пин хранящий значение реле номер один;

– *relay_pin_value*2 – пин хранящий значение реле номер два;

– *gpsd* – устанавливает *gps* мод;

– *pwm*1 – поле для установки начального значения (ШИМ) первого мотор-колеса;

– *pwm*2 – поле для установки начального значения (ШИМ) второго мотор-колеса.

Далее будут рассмотрены методы класса *robotController* для управления движением колесного робота.

Метод *setup* – задает начальные значения полей.

Метод *set_speed* – устанавливает значения скоростей движения мотор-колес и принимает следующие параметры:

– *speed*1 – скорость первого мотор-колеса;

– *speed*2 – скорость второго мотор-колеса.

Метод *stop_motors* – останавливает мотор-колеса.

Метод *turn_left* – осуществляет поворот колесного робота в левую сторону и принимает в себя параметр *speed* – значение скорости.

Метод *turn_right* – осуществляет поворот колесного робота в правую сторону и принимает в себя параметр *speed* – значение скорости.

Метод *get_gps_coordinates* – получает текущие значения *gps* координат.

Метод *move_to_coordinate* – сверяет текущие значения координаты с полученными координатами от клиентской части программного обеспечения и запускает движение робота в заданном направлении.

Метод *move_to_coordinate* принимает следующие параметры:

– *target_latitude* – значение целевой широты, полученной с клиентской части программного обеспечения;

– *target_longitude* – значение целевой долготы, полученной с клиентской части программного обеспечения.

Метод *move_to_target_points* – задает движение робота по заданным координатам.

Метод *move_to_target_points* принимает параметром массив целевых координат, к которым осуществляет движение.

Метод *start* – запускает серверную часть программного обеспечения движения колесного робота по заданной траектории.

Листинг кода представлен в приложении А.

4 ВЕРИФИКАЦИЯ ПРОГРАММНЫХ СРЕДСТВ ОБЕСПЕЧЕНИЯ ДВИЖЕНИЯ АГРАРНОГО РОБОТА ПО ЗАДАННОЙ ТРАЕКТОРИИ

4.1 Алгоритм работы клиентской части программных средств обеспечения движения колесного робота по заданной траектории

После запуска клиентской части программных средств обеспечения движения колесного робота по заданной траектории открывается веб-страница в браузере на ПК с картой для формирования маршрута, с некоторым функционалом.

Графический интерфейс клиентской части программных средств представлен на рисунке 4.1.

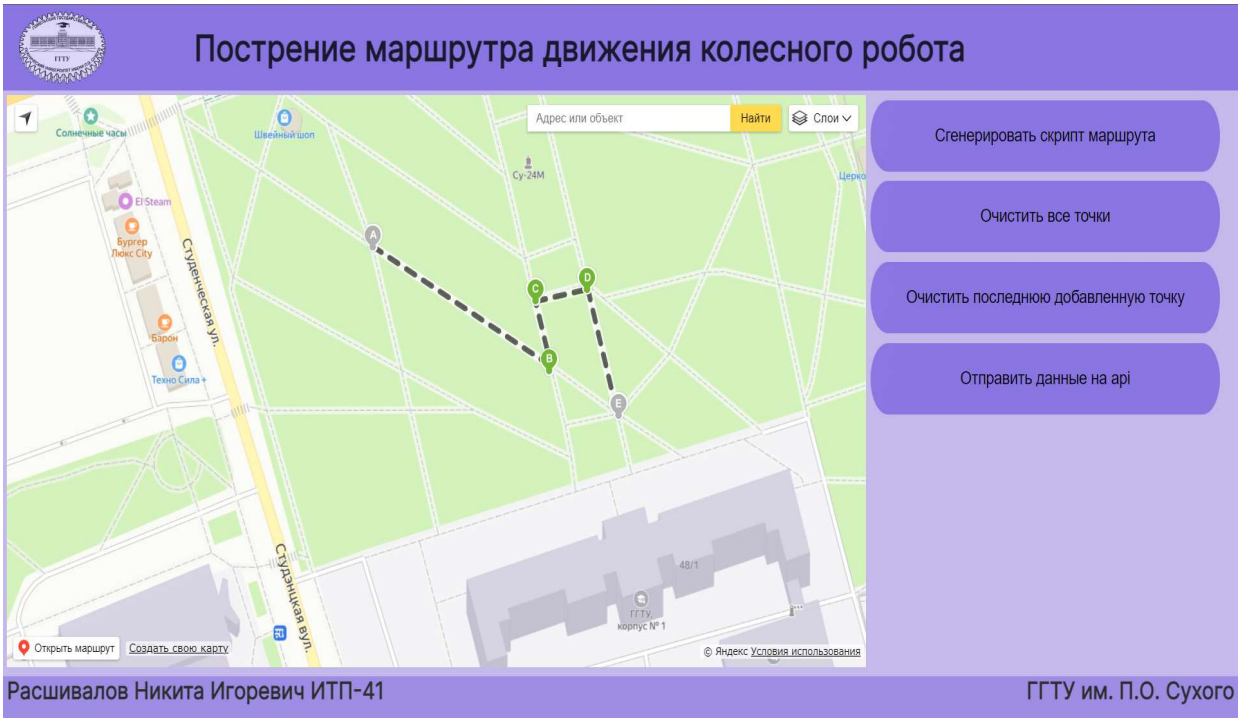

Рисунок 4.1 – Графический интерфейс клиентской части программных средств

Для формирования маршрута необходимо выбрать на карте точки по которым будет двигаться колесный робот.

После выбора конечных точек при нажатии на кнопку «Очистить все точки» карта будет очищена – то есть удалены все точки маршрута движения колесного робота.

При необходимости изменить построенный маршрут или удалить последнюю добавленную точку, необходимо нажать кнопку «Удалить последнюю добавленную точку», после которой маршрут будет изменен.

При нажатии на кнопку «Сгенерировать скрипт маршрута» произойдет генерация скрипта, содержащего набор конечных точек следования колесного робота.

При нажатии на кнопку «Отправить данные на *API*» происходит генерация скрипта и отправка его на серверную часть программных средств обеспечения движения колесного робота по заданной траектории.

На рисунке 4.2 показан сгенерированный скрипт маршрута.

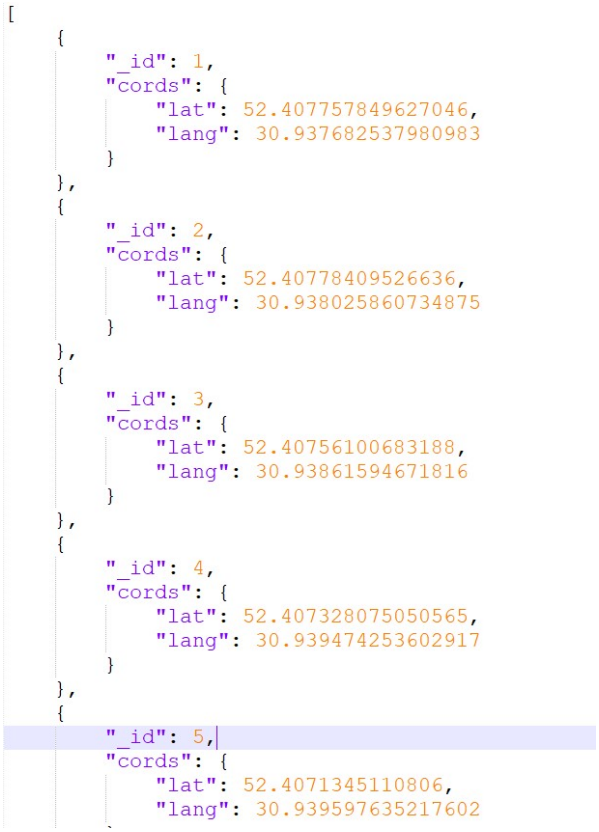

Рисунок 4.2 – Скрипт сгенерированного маршрута

Скрипт, сгенерированный на клиентской части, содержит координаты долготы и широты, а также порядковые номера точек на маршруте.

4.2 Алгоритм работы серверной части программных средств обеспечения движения колесного робота по заданной траектории

После включения колесного робота на нем срабатывает скрипт, обеспечивающий движение по заданной траектории.

Заданный маршрут генерируется на клиентской части программных средств обеспечения движения колесного робота и отправляется на серверную часть ПО.

Если между серверной и клиентской частью не удается установить соединение или же соединение с сетью интернет отсутствует, то скрипт маршрута будет читаться с локального файла, лежащего в папке с исполняемым скриптом.

Если же скрипт маршрута отсутствует, то его необходимо вручную импортировать на устройство или же найти способ подключиться к сети интернет.

После прочтения файла, содержащего скрипт с построенным маршрутом, робот начнет двигаться в заданном направлении.

На рисунке 4.3 изображен колесный робот с загруженным маршрутом и готовый начать движение по заданной траектории.

Рисунок 4.3 – Колесный робот с загруженным маршрутом

На рисунке 4.4 показан скрипт, полученный с клиентской части приложения.

```
[Coords(id=1, lat=52.40647541559637, lng=30.936403957524494), Coords(id=2, lat=52.405852058837375, lng=30.9368760263111
2), Coords(id=3, lat=52.40518276067273, lng=30.937187162556857), Coords(id=4, lat=52.40492684985396, lng=30.93827077499
877)]
```

Рисунок 4.4 – скрипт, полученный с клиентской части программного обеспечения

Скрипт содержит в себе данные долготы и широты всех выбранных точек маршрута и порядковый номер координаты.

4.3 Тестирование программного продукта

Для тестирования разработанного программного продукта используются разные подходы в зависимости от компонента.

4.3.1 Модульное-тестирование – модульные тесты представляют собой специальные тестовые сценарии, которые проверяют корректность работы отдельных модулей или компонентов программы, называемых юнитами, создаются во время разработки кода и проверяют, что каждая функция или метод возвращает ожидаемый результат при заданных входных данных.

Модульное-тестирование используется для проверки кода, написанного на *Python* т.к. данный код не имеет внешних зависимостей и легко поддается данному виду тестирования.

Для реализации тестов используется встроенный модуль *unittest*, предоставляющий функциональность для создания и запуска тестовых сценариев. Он предоставляет набор инструментов для автоматической проверки функций, методов и классов на соответствие ожидаемому поведению. Для тестирования создан отдельный проект, в котором помещаются классы с тестами и данные необходимые при тестировании.

Модульное тестирование состоит из двух проектов:

– *testClientData* – проверяет взаимодействие с клиентской частью;

– *testRobotController* – проверяет функции управления роботом.

В *testClientData* проверяется функциональность и поведение приложения, основанного на фреймворке *Flask* и использующего *SocketIO* для обмена данными с клиентской стороной.

В проекте проверяются следующие тестовые случаи:

– *test_connection*: проверяет, что клиент успешно подключается к серверу. Проверяется наличие события «*connect*» в ответе от сервера;

– *test_disconnection*: проверяет, что клиент успешно отключается от сервера. Проверяется наличие события «*disconnect*» в ответе от сервера;

– *test_handle_data_invalid*: проверяет обработку недопустимых данных клиентом. В данном случае, отправляется неверный *JSON* и проверяется, что сервер отправляет сообщение об ошибке и ответное сообщение;

– *test_handle_data_valid*: проверяет обработку корректных данных клиентом. В данном случае, отправляется корректный *JSON* и проверяется, что сервер отправляет ответное сообщение;

– *test_parseData*: проверяет правильность обработки данных сервером. В этом тесте отправляются данные для обработки, и затем проверяется, что переменная *pointsToMove* содержит правильно обработанные данные.

Тестирование включает проверку основных функциональных возможностей приложения, таких как подключение и отключение клиента, обработка данных и корректность работы функций парсинга и обновления данных.

В *testRobotController* тестируется функциональность и поведение класса *RobotController*, который отвечает за управление роботом и взаимодействие с *GPS*-модулем.

В проекте проверяются следующие тестовые случаи:

– *test_set_speed:* проверяет, что метод *set_speed* правильно устанавливает скорость двигателей робота путем изменения состояния объектов *PWM*;

– *test_stop_motors:* проверяет, что метод *stop_motors* останавливает двигатели робота путем установки состояния *PWM* в нуль;

– *test_turn_left*: Проверяет, что метод *turn_left* поворачивает робота влево путем изменения состояния объекта *PWM* для соответствующего двигателя;

– *test_turn_right:* проверяет, что метод *turn_right* поворачивает робота вправо путем изменения состояния объекта *PWM* для соответствующего двигателя;

– *test_get_gps_coordinates_valid*: проверяет, что метод *get_gps_coords* возвращает правильные координаты, когда объект *GPS*-модуля содержит валидные данные;

– *test_get_gps_coordinates_invalid:* проверяет, что метод *get_gps_coords* возвращает *None*, когда объект *GPS*-модуля содержит невалидные данные;

– *test_move_to_coordinate*: проверяет, что метод *move_to_coordinate* правильно определяет направление движения робота на основе текущих и целевых координат;

– *test_move_to_target_points*: проверяет, что метод *move_to_target_points* последовательно вызывает *move_to_coordinate* для каждой целевой координаты из списка;

– *test_start*: проверяет основной цикл работы робота в методе *start*, включая обработку сигналов реле и вызов методов управления двигателями и перемещения к целевым координатам.

Тестирование включает проверку различных аспектов работы класса *RobotController*, включая установку скорости, остановку и поворот двигателей, получение *GPS*-координат, перемещение к указанным координатам и обработку сигналов реле.

Модульное тестирование позволяет обнаружить потенциальные проблемы и ошибки в коде, а также подтвердить, что каждый компонент работает правильно в изоляции. Это помогает обеспечить более надежное и стабильное функционирование приложения.

4.3.2 Тестирование работы удаленного соединения с одноплатным компьютером, на котором установлен *VNC* сервер представляет собой проверку соединения между *VNC*-клиентом (предварительно расположенном на ПК) и *VNC*-сервером. Оно включает в себя несколько шагов. Сначала необходимо убедиться, что установлены *VNC*-клиент и *VNC*-сервер. Если нет, загрузить и установить их на соответствующие устройства.

Далее запускается *VNC*-клиент на управляющем устройстве. Вводится *IP*-адрес или имя удаленного компьютера, а также порт, на котором работает *VNC*-сервер. Затем нажимается кнопка «Подключиться», для установления соединения с *VNC*-сервером и появляется сообщение об успешном подключении.

Далее появляется удаленный рабочий стол – графический интерфейс одноплатного компьютера.

На рисунке 4.5 изображен графический интерфейс одноплатного компьютера.

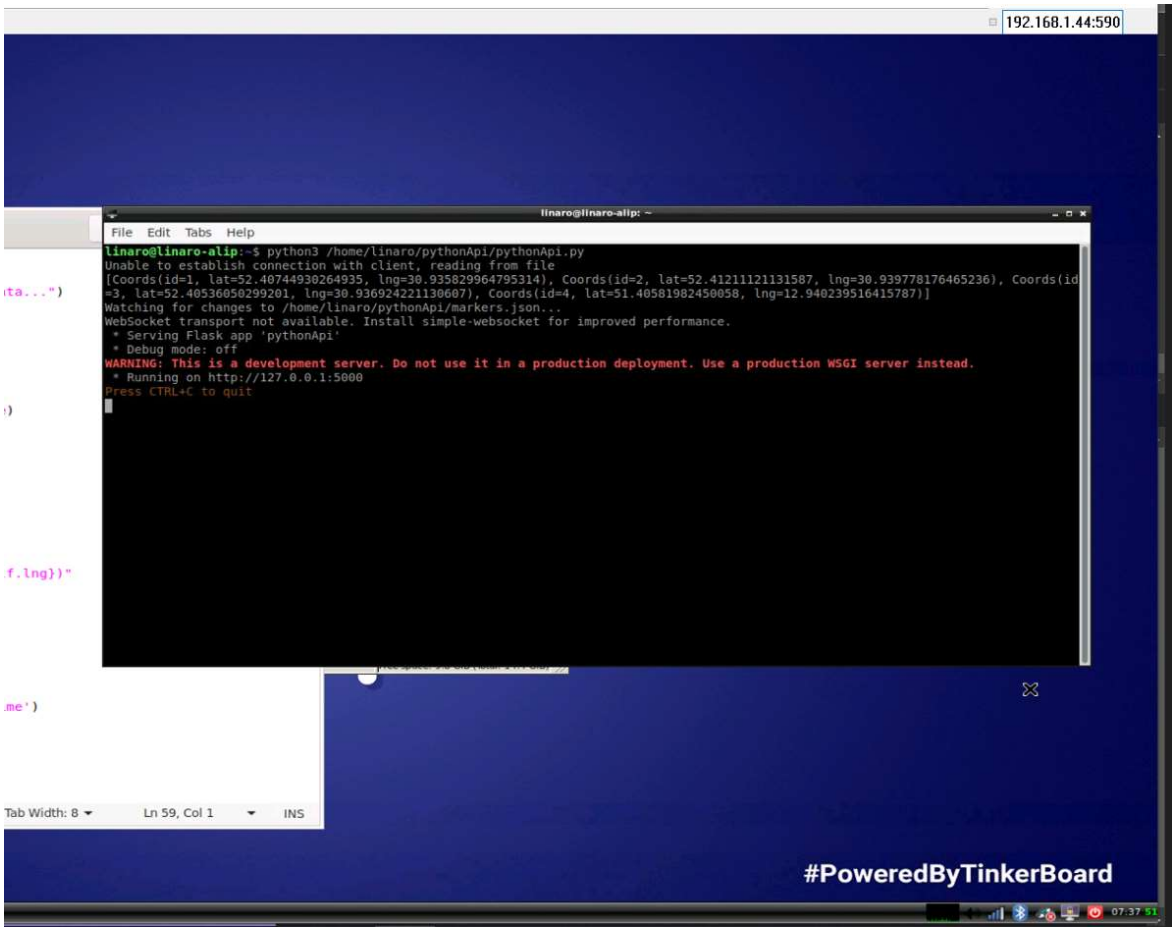

Рисунок 4.5 – Графический интерфейс одноплатного компьютера

После успешного подключения к удаленному рабочему столу проверяется состояние включенных сервисов в системе, где отображается скрипт *RobotController.py*, так как он стоит в автозапуске системы.

Далее вручную изменяется файл *markers.json*, находящийся в одной директории с исполняемым скриптом и хранящие данные о маршруте.

После изменения скрипта подается сигнал на движение по измененному маршруту и начинается движение робота.

4.3.3 Функциональное тестирование – это процесс проверки программного обеспечения с целью убедиться, что система работает в соответствии с заданными функциональными требованиями и ожидаемым поведением. Сосредоточено на проверке функций и возможностей программы в целом, в отличие от юнит-тестирования, которое проверяет отдельные компоненты или модули.

Данный вид тестирования используется для проверки логики написанной на клиентской части программного обеспечения на *React-js* и *Python*.

Целью данного функционального тестирования является проверка корректной работы модуля *sendDataToApi*, который предоставляет функциональность отправки данных через *socket.io-client* и скрипта управления движением робота.

Для тестирования модуля *sendDataToApi* используются инструменты *Jest* и *Enzyme*, позволяющие создавать и запускать тесты.

Тесты организованы в блоке *describe*, который объединяет связанные тесты в одну группу. Это помогает в организации и структурировании тестового кода.

Перед каждым тестом используется функция *beforeEach*, которая выполняет подготовительные действия, такие как создание фиктивного объекта *socket.io-client* с помощью *jest.mock*. Это позволяет контролировать поведение объекта *socket.io-client* и имитировать его методы.

Также перед каждым тестом используется функция *afterEach*, которая выполняет очистку имитаторов с помощью *jest.clearAllMocks*. Это гарантирует, что состояние будет сброшено перед каждым тестом, чтобы не возникало взаимного влияния между ними.

Каждый тест проверяет определенный аспект функциональности модуля *sendDataToApi*. Например, тест с названием «*should send valid data to the server*» проверяет, что при вызове функции *sendDataToApi* с корректными данными, устанавливается соединение с сервером и данные успешно отправляются через сокет. Для этого создается экземпляр *sendDataToApi* с определенными параметрами, а затем вызывается функция *sendData*. После этого проверяются ожидаемые вызовы функций и логирование с помощью специальных функций-ожиданий.

Тесты также проверяют различные сценарии ошибок, такие как ошибки подключения к серверу и ошибки сокета. Они проверяют, что соответствующие сообщения об ошибке правильно логируются с использованием *console.error*.

В целом, функциональное тестирование модуля *sendDataToApi* позволяет убедиться в его надежности и корректной работе в различных сценариях использования. Это способствует повышению качества разработки и обеспечению безопасности передачи данных через сокеты.

Также происходит тестирование модуля, установленного на колесном роботе.

Колесный робот с корректно подключенным одноплатным компьютером и всеми дополнительными устройствами включая контроллеры, преобразователи и реле изображен на рисунке 4.6.

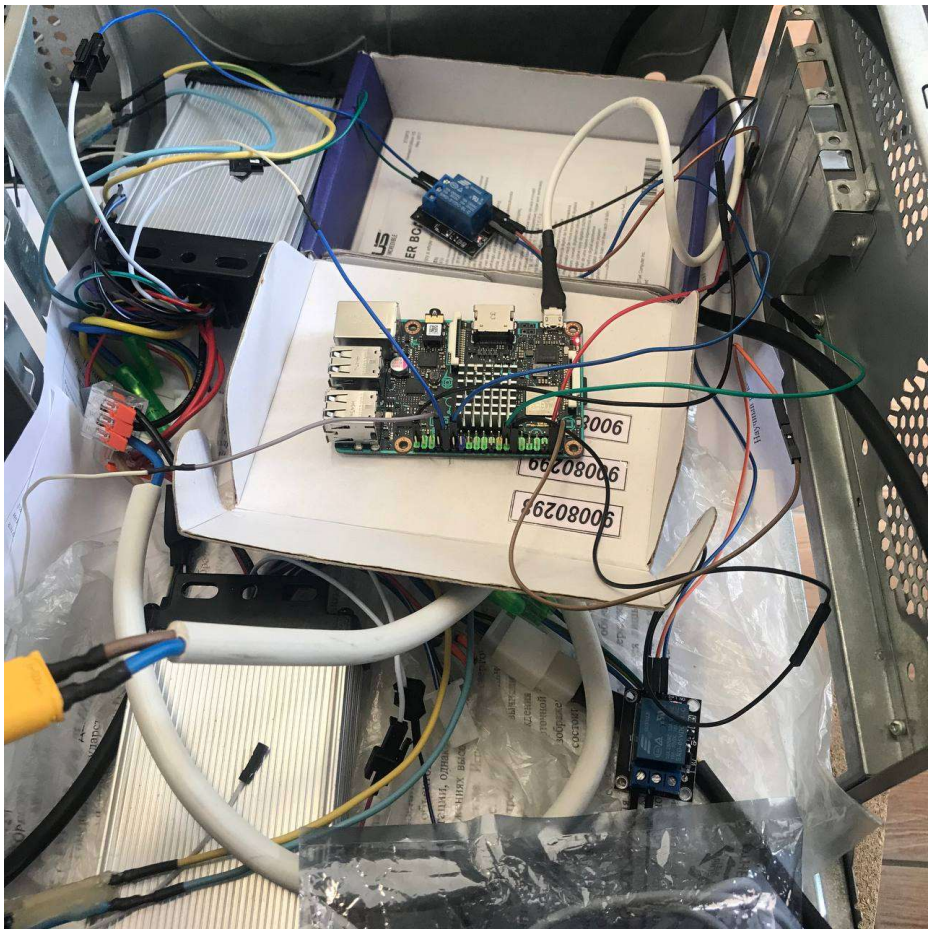

Рисунок 4.6 – Колесный робот с корректно подключенными устройствами

Для тестирования скрипта управления колесным роботом были проведены соответствующие тесты.

Тест № 1. Цель: проверить запуск приложения.

Ожидаемый результат: скрипт должен загрузиться, проверка на соединение между серверной частью и клиентской должна быть пройдена.

Вывод: приложение запускается и начинает работу без ошибок.

Тест № 2. Цель: проверить отправку данных и обработку их на сервере.

Ожидаемый результат: клиент отправляет серверу корректные данные, сервер успешно принимает эти данные и обрабатывает.

Вывод: ожидаемый и полученный результаты совпадают, цель достигнута.

Тест № 3. Цель: проверить чтение локального скрипта на роботе в связи с отсутствием интернет соединения.

Ожидаемый результат: серверный скрипт должен читать координаты точек маршрута из локального файла.

Вывод: скрипт читает данные из локального файла, ожидаемый и полученный результат совпадают, цель достигнута.

Тест № 4. Цель: проверить процесс запуска мотор-колес.

Ожидаемый результат: при подаче сигнала на движение, происходит вращение колес.

Вывод: при подаче сигнала мотор-косела начинают вращаться с заданной скоростью, ожидаемый и полученный результаты совпадают, цель достигнута.

Тест № 5. Цель: проверить процесс остановки мотор-колес.

Ожидаемый результат: при подаче сигнала на остановку, прекращается вращение колес.

Вывод: при подаче сигнала на остановку мотор-колеса прекращают вращаться, ожидаемый и полученный результаты совпадают, цель достигнута.

Тест № 6. Цель: проверить процесс поворота робота.

Ожидаемый результат: при подаче сигнала на поворот в правую или левую стороны, или разворот, робот производит заданные действия.

Вывод: при подаче сигнала на поворот(разворот) робот производит заданные действия, ожидаемый и полученный результаты совпадают, цель достигнута.

Тест № 7. Цель: проверить следование по маршруту.

Ожидаемый результат: при чтении скрипта, содержащего маршрут, робот проезжает по заданной траектории.

Вывод: при чтении скрипта, содержащего маршрут, робот начинает движение по заданной траектории, ожидаемый и полученный результаты совпадают, цель достигнута.

5 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ПРОГРАММНЫХ СРЕДСТВ ОБЕСПЕЧЕНИЯ ДВИЖЕНИЯ АГРАРНОГО РОБОТА ПО ЗАДАННОЙ ТРАЕКТОРИИ

5.1 Технико-экономическое обоснование целесообразности разработки программного продукта и оценка его конкурентоспособности

Техническая прогрессивность разрабатываемого программного продукта определяется коэффициентом эквивалентности ($К_{эк}$). Расчет этого коэффициента осуществляется путем сравнения технического уровня товара и разрабатываемого программного продукта по отношению к эталонному уровню программного продукта данного направления с использованием формулы (Д.1). Результат расчета коэффициента эквивалентности приведен в таблице 5.1. Полученное значение коэффициента эквивалентности больше единицы, следовательно, разрабатываемый программный продукт является технически прогрессивным.

Таблица 5.1 – Расчет коэффициентов эквивалентности

Наименование параметра	Вес параметра, β	Значения параметра			$\frac{P_б}{P_э}$	$\frac{P_н}{P_э}$	$\beta\frac{P_б}{P_э}$	$\beta\frac{P_н}{P_э}$
		$P_б$	$P_н$	$P_э$				
Объем памяти	0,1	9	5	6	1,5	0,83	0,15	0,08
Время обработки	0,4	0,7	0,5	0,2	3,5	2,5	1,4	1
Отказы	0,5	2	1	1	2	1	1	0,5
Итого							2,15	1,58
Коэффициент эквивалентности							2,15/1,58=1,36	

Далее рассчитывается коэффициент изменения функциональных возможностей ($К_{ф.в}$) нового программного обеспечения по формуле (Д.3). Расчет коэффициента изменения функциональных возможностей нового программного продукта приведен в таблице 5.2.

Таблица 5.2 – Расчет коэффициента изменения функциональных возможностей

Наименование показателя	Балльная оценка базового ПП	Балльная оценка нового ПП
1	2	3
Объем памяти	3	4

Продолжение таблицы 5.2

1	2	3
Быстродействие	3	4
Удобство интерфейса	3	4
Степень утомляемости	2	1
Производительность труда	3	4
Итого	14	19
Коэффициент функциональных возможностей	17/14 = 1,21	

Новый программный продукт превосходит по своим функциональным возможностям базовый в 1,21 раза.

Конкурентоспособность нового программного продукта по отношению к базовому можно оценить с помощью интегрального коэффициента конкурентоспособности, учитывающего все ранее рассчитанные показатели. Для расчета конкурентоспособности формула (Д.4).

Коэффициент цены потребления рассчитывается как отношение договорной цены нового программного продукта к договорной цене базового (таблица 5.3).

Таблица 5.3 – Расчет уровня конкурентоспособности нового ПП

Коэффициенты	Значение
Коэффициент эквивалентности	1,36
Коэффициент изменения функциональных возможностей	1,21
Коэффициент соответствия нормативам	1
Коэффициент цены потребления	0,94
Интегральный коэффициент конкурентоспособности	$(1{,}36 \cdot 1{,}27 \cdot 1)/0{,}93 = 1{,}85$

Интегральный коэффициент конкурентоспособности ($К_и$) больше единицы, это значит, что новый программный продукт является более конкурентоспособным, чем базовый.

5.2 Оценка трудоемкости работ по созданию программного обеспечения

Общий объем программного обеспечения (V_o)) определяется исходя из количества и объема функций, реализуемых программой, по каталогу функций программного обеспечения по формуле (Д.5).

Уточненный объем программного обеспечения (V_y)) определяется по формуле (Д.6).

Результаты произведённых вычислений объема функций ПО представлены в таблице 5.4.

Таблица 5.4 – Перечень и объем функций ПО

Код функции	Наименование (содержание) функции	Объем функции строк исходного кода	
		по каталогу (V_o))	Уточненный (V_y))
101	Организация ввода информации	150	170
102	Контроль, предварительная обработка и ввод информации	450	460
109	Управление вводом-выводом	1200	730
506	Обработка ошибочных сбойных ситуаций	1000	900
702	Расчетные задачи	3000	2450
707	Графический вывод результатов	500	420
709	Изменение состояния ресурсов в интерактивном режиме	520	400
Итого		6820	5530

ПО относится к третьей категории сложности.

На основании принятого к расчету (уточненного) объема (V_y) и категории сложности ПО определяется нормативная трудоемкость ПО ($T_н$) выполняемых работ, представлена в таблице 5.5.

Таблица 5.5 – Нормативная трудоемкость на разработку ПО ($T_н$)

Уточнённый объем, V_y	1-я категория сложности ПО	Номер нормы
5530	340	51

Дополнительные затраты труда, связанные с повышением сложности разрабатываемого ПО, учитываются посредством коэффициента повышения сложности ПО ($К_с$). $К_с$ рассчитывается по формуле (Д.7):

$$K_с = 1 + 0,08 = 1,08.$$

Влияние фактора новизны на трудоемкость учитывается путем умножения нормативной трудоемкости на соответствующий коэффициент, учитывающий новизну ПО ($К_н$).

Разработанная программа обладает категорией новизны Б, а значение $К_н$ = 0,72.

Степень использования в разрабатываемом ПО стандартных модулей определяется их удельным весом в общем объеме ПО.

В данном программном комплексе используется до 50 процентов стандартных модулей, что соответствует значению коэффициента $К_т$ = 0,65.

Программный модуль разработан с помощью объектно-ориентированных технологий, что соответствует коэффициенту, учитывающему средства разработки ПО, $К_{у.р}$ = 0,6. Значения коэффициентов удельных весов трудоемкости стадий разработки ПО в общей трудоемкости ПО определяются с учетом установленной категории новизны ПО и приведены в таблице 5.6.

Таблица 5.6 – Значения коэффициентов удельных весов трудоемкости стадий разработки ПО в общей трудоемкости

Категория новизны ПО	Без применения *CASE*-технологий				
	Стадии разработки ПО				
	ТЗ	ЭП	ТП	РП	ВН
	Значения коэффициентов				
	$К_{т.з}$	$К_{э.п}$	$К_{т.п}$	$К_{р.п}$	$К_{в.н}$
Б	0,10	0,20	0,30	0,30	0,10

Нормативная трудоемкость ПО ($Т_н$) выполняемых работ по стадиям разработки корректируется с учетом коэффициентов: повышения сложности ПО ($К_с$), учитывающих новизну ПО ($К_н$), учитывающих степень использования стандартных модулей ($К_т$), средства разработки ПО ($К_{у.р}$) и определяются для стадии ТЗ по формуле (Д.8), для стадии ЭП по формуле (Д.9), для стадии ТП по формуле (Д.10), для стадии РП по формуле (Д.11), для стадии ВН по формуле (Д.12). Коэффициенты $К_н$, $К_с$ и $К_{у.р}$ вводятся на всех стадиях разработки, а коэффициент $К_т$ вводится только на стадии РП.

Для уменьшения общей трудоёмкости разработки введем коэффициент используемости разработанного ПО, который равен 0,1.

$$T_{у.т.з} = 340 \cdot 0,1 \cdot 1,08 \cdot 0,72 \cdot 0,6 = 16 \text{ чел.-дн.},$$

$$T_{у.э.п} = 340 \cdot 0,2 \cdot 1,08 \cdot 0,72 \cdot 0,6 = 33 \text{ чел.-дн.},$$

$$T_{у.т.п} = 340 \cdot 0,3 \cdot 1,08 \cdot 0,72 \cdot 0,6 = 48 \text{ чел.-дн.},$$

$$T_{у.р.п} = 340 \cdot 0,3 \cdot 1,08 \cdot 0,72 \cdot 0,65 \cdot 0,6 = 31 \text{ чел.-дн.},$$

$$T_{у.в.н} = 340 \cdot 0,1 \cdot 1,08 \cdot 0,72 \cdot 0,6 = 16 \text{ чел.-дн.}$$

Общая трудоемкость разработки программного обеспечения ($T_о$) определяется суммированием нормативной (скорректированной) трудоемкости программного обеспечения по стадиям разработки по формуле (Д.13):

$$T_о = 16 + 33 + 48 + 31 + 16 = 144 \text{ чел.-дн.}$$

Параметры расчетов по определению нормативной и скорректированной трудоемкости программного обеспечения по стадиям разработки и общую трудоемкость разработки представлены в таблице 5.7.

Таблица 5.7 – Расчет общей трудоемкости разработки ПО

Показатели	Стадии разработки					Итого
	ТЗ	ЭП	ТП	РП	ВН	
1	2					3
Общий объем ПО ($V_о$), кол-во строк *LOC*	–	–	–	–	–	6820
Общий уточненный объем ПО ($V_у$), кол-во строк *LOC*	–	–	–	–	–	5530
Категория сложности разрабатываемого ПО	–	–	–	–	–	1
Нормативная трудоемкость разработки ПО ($T_н$), чел.-дн.	–	–	–	–	–	340
Коэффициент повышения сложности ПО ($K_с$)	1,08	1,08	1,08	1,08	1,08	–

Продолжение таблицы 5.7

Коэффициент, учитывающий новизну ПО ($К_н$)	0,72	0, 72	0, 72	0, 72	0, 72	–
Коэффициент, учитывающий степень использования стандартных модулей ($К_т$)	–	–	–	0,65	–	–
Коэффициент, учитывающий средства разработки ПО($К_{у.р}$)	0,6	0,6	0,6	0,6	0,6	–
Коэффициенты весов трудоемкости стадий разработки ПО ($К_{т.з}$, $К_{э.п}$, $К_{т.п}$, $К_{р.п}$, $К_{в.н}$)	0,10	0,20	0,30	0,30	0,10	1,0
Распределение скорректированной трудоемкости ПО по стадиям, чел.-дн.	16	33	48	31	16	144
Общая трудоемкость разработки ПО ($Т_о$), чел.-дн.	–	–	–	–	–	144

Таким образом общая трудоемкость разработки составляет 144 чел.-дн.

5.3 Расчет затрат на разработку программного продукта

Суммарные затраты на разработку программного обеспечения ($З_р$) определяются по формуле (Д.14). Параметры расчета производственных затрат на разработку программного обеспечения приведены в таблице 5.8.

Таблица 5.8 – Параметры расчета производственных затрат на разработку ПО

Параметр	Единица измерения	Значение
Тарифная ставка	руб.	554
Доплата по контракту	%	0,2 от оклада
Норматив отчислений на доп. зарплату разработчиков ($Н_{доп}$)	%	20
Численность обслуживающего персонала	чел.	1
Средняя годовая ставка арендных платежей ($С_{ар}$) (по результатам мониторинга предложений по аренде помещений)	руб./м^2	16,9

Продолжение таблицы 5.8

Площадь помещения (S)	$м^2$	15
Количество ПЭВМ ($Q_{эвм}$)	шт.	1
Затраты на приобретение единицы ПЭВМ	руб.	2500
Стоимость одного кВт-часа электроэнергии ($С_{эл}$)	руб.	0,33
Коэффициент потерь рабочего времени ($К_{пот}$)	–	0,2
Норматив общепроизводственных затрат ($Н_{общ.пр}$)	%	10
Норматив непроизводственных затрат ($Н_{непр}$)	%	5

Расходы на оплату труда разработчиков с отчислениями ($З_{т.р}$) определяются по формуле (Д.15).

Основная заработная плата разработчиков рассчитывается по формуле (Д.16).

Средняя часовая тарифная ставка определяется по формуле (Д.17).

Часовая тарифная ставка определяется путем деления месячной тарифной ставки на установленный при восьмичасовом рабочем дне фонд рабочего времени 168 ч ($F_{мес}$), формула (Д.18).

$$С_{ср.час} = \frac{554 + 0{,}2 \cdot 554}{168} = 3{,}95 \text{ руб./ч}$$

$$ЗП_{осн} = 3{,}95 \cdot 144 \cdot 1{,}8 \cdot 8 = 8190 \text{ руб.}$$

Дополнительная заработная плата рассчитывается по формуле (Д.19):

$$ЗП_{доп} = 8190 \cdot 0{,}2 = 1638 \text{ руб.}$$

Отчисления от основной и дополнительной заработной платы рассчитываются по формуле (Д.20):

$$ОТЧ_{с.н} = \frac{(8190 + 1638) \cdot 34}{100} = 3341 \text{ руб.}$$

$$З_{тр} = 8190 + 1638 + 3341 = 13169 \text{ руб.}$$

Годовые затраты на аренду помещения определяются по формуле (Д.27):

$$З_{ар} = 16,9 \cdot 15 = 253,5 \text{ руб.}$$

Сумма годовых амортизационных отчислений ($З_{ам}$) определяется по формуле (Д.28):

$$З_{ам} = 2500 \cdot (1 + 0,13) \cdot 0,125 = 354 \text{ руб.}$$

Стоимость электроэнергии, потребляемой за год, определяется по формуле (Д.29).

Действительный годовой фонд времени работы ПЭВМ ($F_{эвм}$) рассчитывается по формуле (Д.30):

$$F_{эвм} = (365 - 113) \cdot 8 \cdot 1 \cdot (1 - 0,2) = 1612,8 \text{ ч.}$$

$$З_{э.п} = \frac{0,41 \cdot 1612,8 \cdot 0,33 \cdot 0,9}{1} = 196,4 \text{ руб.}$$

Затраты на материалы ($З_{в.м}$), необходимые для обеспечения нормальной работы ПЭВМ, составляют около один процент от балансовой стоимости ЭВМ, и определяются по формуле (Д.31):

$$З_{в.м} = 2500 \cdot (1 + 0,13) \cdot 0,01 = 28,25 \text{ руб.}$$

Затраты на текущий и профилактический ремонт ($З_{т.р}$) принимаются равными пять процентов от балансовой стоимости ЭВМ и вычисляются по формуле (Д.32):

$$З_{т.р.} = 2500 \cdot (1 + 0,13) \cdot 0,05 = 141,2 \text{ руб.}$$

Прочие затраты, связанные с эксплуатацией ЭВМ ($З_{пр}$), состоят из амортизационных отчислений на здания, стоимости услуг сторонних организаций и составляют пять процентов от балансовой стоимости и определяются по формуле (Д.33):

$$З_{пр} = 2500 \cdot (1 + 0,13) \cdot 0,05 = 141,2 \text{ руб.}$$

Для расчета машинного времени ЭВМ ($t_{эвм}$ в часах), необходимого для разработки и отладки проекта специалистом, следует использовать формулу (Д.34):

$$t_{эвм} = 47 \cdot 8 \cdot 1 = 376 \text{ ч.}$$

$$C_ч = \frac{253,5 + 354 + 196,4 + 28,25 + 141,2 + 141,2}{1644,8} = 0,67 \text{ руб./ч.}$$

$$З_{мв} = 0,67 \cdot 367 = 252 \text{ руб.}$$

Расчет затрат на изготовление эталонного экземпляра ($З_{эт}$) осуществляется по формуле (Д.35):

$$З_{эт} = (13169 + 252) \cdot 0,05 = 671 \text{ руб.}$$

Таким же образом, рассчитываются затраты на материалы ($З_{мат}$) по формуле (Д.36):

$$З_{мат} = 2500 \cdot (1 + 0,1) \cdot 0,01 = 27,5 \text{ руб.}$$

Общепроизводственные затраты ($З_{общ.пр}$) рассчитываются по формуле (Д.37):

$$З_{общ.пр} = \frac{8190 \cdot 10}{100} = 819 \text{ руб.}$$

И наконец, непроизводственные затраты рассчитываются по формуле (Д.38):

$$З_{непр} = \frac{8190 \cdot 5}{100} = 410 \text{ руб.}$$

Итого получаем суммарные затраты на разработку:

$$З_р = 13169 + 252 + 671 + 27,5 + 819 + 410 = 15349 \text{ руб.}$$

5.4 Расчет договорной цены и частных экономических эффектов от производства и использования программного продукта

Оптовая цена программного продукта ($Ц_{опт}$) определяется по формулам (Д.39) и (Д.40):

$$П_р = \frac{15349 \cdot 30}{100} = 4604 \text{ руб.}$$

$$Ц_{опт} = 15349 + 4604 = 19953 \text{ руб.}$$

Прогнозируемая отпускная цена ПП рассчитывается по формуле (Д.41). Налог на добавленную стоимость ($Р_{ндс}$) рассчитывается по формуле (Д.42):

$$Р_{ндс} = \frac{(15349 + 4604) \cdot 20}{100} = 3990 \text{ руб.},$$

$$Ц_{опт} = 15349 + 4604 + 3990 = 23943 \text{ руб.}$$

Годовой экономический эффект от производства нового программного обеспечения ($Э_{пр}$) определяется по формуле (Д.43).

После просмотра аналогов ПП можно сделать вывод, что цена на разработку коррелирует в диапазоне от 23000 до 43000 руб., значит в среднем цена на аналогичный базовый продукт составляет 33000 руб.

Расчет годового экономического эффекта от производства нового программного продукта представлен в таблице 5.9.

Таблица 5.9 – Расчет годового экономического эффекта от производства нового программного продукта

Наименование параметра	Условн. обознач.	Базовый вариант	Новый вариант
1	2	3	4
Оптовая цена, руб.	$Ц_i$	33000	23943
Норматив рентабельности	R_i	0,15	0,15
Себестоимость производства, руб.	$С_{прi}$	33000/(1+0,15) = = 28695	20820
Удельные капитальные вложения, руб.	$К_р$	2500	
Нормативный коэффициент капитальных вложений	$Е_н$	0,15	0,15
Расчет			
Удельные приведенные затраты на производство ПО, руб.	$З_{прi}$	28695+2500 ·0,15= = 29070	21195
Годовой экономический эффект от производства нового ПП, руб.	$Э_{пр}$	29070–21195= = 7875	
Прирост прибыли, руб.	$\Delta П_{пр}$	(23943 – 20820) – (33000– 28695) = 1182	

В таблице 5.10 представлены значения прибыли, рентабельности.

Таблица 5.10 – Параметры эффективности проекта

Наименование параметра	Значение
Прибыль, руб.	4604
Уровень рентабельности, %	30
Годовой экономический эффект от производства нового ПП, руб.	7875

Срок окупаемости программного продукта (формула Д.44):

$$-23943+\frac{7875}{1+0,085}+\frac{7875}{(1+0,085)^2}+\frac{7875}{(1+0,085)^3}+\frac{7875}{(1+0,085)^4}=1852,3$$

Следовательно, можно сделать вывод о том, что ПП окупится на четвертом году.

Таким образом, по результатам проведенной оценки, установлено, что реализация проекта обоснована и является экономически целесообразной. Об этом свидетельствуют экономический эффект от производства нового программного продукта ($Э_{пр}$ = 7875 руб.). Все данные приведены в итоговой таблице Д.1

.

6 ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ

6.1 Расчет системы заземления подстанции напряжением 10/0,4кВ

Для расчета системы заземления подстанции напряжением 10/0,4 кВ требуется учесть несколько факторов, таких как тип грунта, сопротивление заземляющего устройства и размеры подстанции [9].

Основные шаги, которые следует выполнить при расчете:

– определить тип грунта, на котором будет установлена подстанция. Разные типы грунта имеют разные значения удельного сопротивления;

– рассчитать необходимое значение сопротивления заземляющего устройства. Это значение зависит от требований нормативных документов и особенностей конкретного проекта. В общем случае, сопротивление заземления должно быть достаточно низким, чтобы обеспечить безопасность системы и уровень потенциала земли;

– определить количество и расположение заземляющих электродов. Для достижения требуемого значения сопротивления заземления может потребоваться установка нескольких электродов в определенных местах;

– рассчитать длину и сечение медных или стальных проводников, которые будут использоваться для соединения заземляющих электродов с заземляющей петлей или заземляющими колодцами;

Заземляющее устройство подстанции 10/0,4 кВ одновременно используется при напряжениях ниже и выше 1000 В.

Ниже приведена формула сопротивления ЗУ (6.1):

$$R = \frac{125}{I}, \tag{6.1}$$

где I – расчётный ток замыкания на землю.

Согласно правилам устройства электроустановок, сопротивление ЗУ должно быть не более полученного значения по формуле (6.1).

Ниже приведена формула для расчета силы тока (6.2):

$$I = U_{\text{н}} \cdot \left(\frac{l_{\text{в}}}{350} + \frac{l_{\text{к}}}{10} \right), \tag{6.2}$$

где $U_{\text{н}}$ = 10 кВ – номинальное напряжение;

$l_{\text{в}}$ – длина воздушных линий;

$l_{\text{к}}$ – длина кабелей.

ВЛ и кабели электрически соединены между собой (отходят от общих шин).

В данном случае $l_к$ равна нулю, а общая длина ВЛ 10 кВ, отходящих от подстанции 35/10 кВ, составляет 230 км.

Тогда сила тока равна:

$$I = \frac{10 \cdot 230}{350} = 6{,}6 \text{ А}.$$

Сопротивление соответственно равно:

$$R = \frac{125}{6{,}6} = 18{,}9 \text{ Ом}.$$

К заземляющему устройству на подстанции 10/0,4 кВ присоединяется и нейтраль трансформатора 10/0,4 кВ. Поэтому согласно ПУЭ, сопротивление этих ЗУ должно быть не более 4 Ом. Это сопротивление должно быть обеспечено с учётом использования естественных заземлителей (в данном случае их нет), а также заземлителей повторных заземлений нулевого провода ВЛ 0,38 кВ (количество ВЛ не менее двух).

Сопротивление заземлителя, расположенного в непосредственной близости от нейтрали трансформатора, должно быть не более 30 Ом (при линейном напряжении 380 В).

Удельное сопротивление земли r более 100 Ом · м допускает увеличение этих норм в 0,01 r раз, но не более десятикратного.

Подобный расчёт заземления подстанции 10/0,4 кВ номер два с тремя отходящими линиями при условии, что ВЛ номер один находится в ремонте. Тогда на 2-х других ВЛ число повторных заземлений нулевого провода равно 15, а их общее сопротивление 2 Ом.

Таким образом, при учёте повторных заземлений обеспечивается величина сопротивления ЗУ $R < 4$ Ом.

Однако, как уже отмечалось, в непосредственной близости от нейтрали трансформатора должен находиться заземлитель с сопротивлением не более 30 Ом (при удельном сопротивлении грунта $r = 100$ Ом · м). Так как 30 Ом >18,5 Ом (предельная величина сопротивления ЗУ по величине тока замыкания на землю), то на подстанции необходимо выполнить ЗУ с сопротивлением $R \leq 18{,}9$ Ом.

Заземляющее устройство выполняется в виде прямоугольного контура из горизонтально проложенной на глубине 0,8 м круглой стали диаметром 10 мм и из расположенных по этому контуру вертикальных стержней из угловой стали

40 · 40 · 4 мм длиной $l_в$ = 3м, отстоящих друг от друга на одинаковом расстоянии а = 3 м. Удельное сопротивление земли r = 60 Ом · м.

Ниже приведена формула расчётного значения удельного сопротивления грунта (6.3):

$$r_r = K \cdot r, \tag{6.3}$$

где r – удельное сопротивление земли;

K – коэффициент сезона.

Тогда расчётное значение удельного сопротивления грунта составит r_r = 90 Ом · м для вертикальных стержней и 132 Ом · м для горизонтальных заземлителей.

Ниже представлена формула сопротивления одного стержня из угловой стали, верхний конец которого находится на глубине до 0,8 м (6.4):

$$R_в = 0{,}366 \frac{r_r}{l} \cdot \left(\lg \frac{zl}{0{,}95} + 0{,}5 \lg \frac{4t + 3l}{4t + l} \right), \tag{6.4}$$

где l – длина стержня.

Тогда сопротивление одного стержня из угловой стали, верхний конец которого находится на глубине до 0,8 м:

$$R_в = 0{,}366 \frac{90}{3} \cdot \left(\lg \frac{2 \cdot 3}{0{,}95 \cdot 0{,}04} + 5 \lg \frac{4 \cdot 0{,}8 + 3 \cdot 3}{4 \cdot 0{,}8 + l} \right) = 25 \text{ Ом}.$$

Ориентировочное число вертикальных стрежней без учёта их взаимного экранирования находится по формуле (6.5):

$$n = \frac{R_в}{R}, \tag{6.5}$$

Ориентировочное число вертикальных стрежней без учёта их взаимного экранирования равно:

$$n = \frac{25}{18{,}9} = 1{,}3.$$

Однако со стороны входа на подстанцию для выравнивания потенциала должны располагаться два вертикальных стержня, причём пройти на территорию

подстанции можно как с одной стороны, так и с другой. Поэтому $n = 4$. При $n = 4$ и $\frac{a}{l} = 1$ коэффициент использования вертикальных стержней в контуре $\eta_{в.к.} = 0{,}5$.

Тогда результирующее сопротивление всех вертикальных стрежней с учётом их взаимного экранирования находится по формуле (6.6):

$$R_{в.э.} = \frac{R_в}{n \cdot \eta_{в.к.}}, \tag{6.6}$$

Результирующее сопротивление всех вертикальных стрежней с учётом их взаимного экранирования равно:

$$R_{в.э.} = \frac{25}{4 \cdot 0{,}5} = 12{,}5\,\text{Ом}.$$

Ниже приведена формула длиной горизонтального заземлителя (6.7):

$$l_г = n \cdot a, \tag{6.7}$$

Полученное значение длинны горизонтального заземлителя равно:

$$l_г = 4 \cdot 3 = 12\text{м}.$$

Ниже приведена формула сопротивления горизонтального заземлителя длиной $l_г$ (6.8):

$$R_г = 0{,}366 \frac{r_r}{l_g} \cdot \lg \frac{2{l_г}^2}{2d \cdot t}, \tag{6.8}$$

где d – диаметр заземлителя в метрах;

t – глубина заложения в метрах.

Тогда результирующее сопротивление всех вертикальных стержней равно:

$$R_{г.э.} = \frac{17}{0{,}45} = 38\,\text{Ом}.$$

Ниже приведена формула результирующего сопротивление ЗУ (6.9):

$$R = \frac{R_{\text{в.э.}} \cdot R_{\text{г.э.}}}{R_{\text{в.э.}} + R_{\text{г.э.}}}, \tag{6.9}$$

В конечном итоге результирующее сопротивление ЗУ равно:

$$R = \frac{12{,}5 \cdot 38}{12{,}5 + 38} = 9{,}4\,\text{Ом}.$$

Из расчетов следует, что результирующее сопротивление заземляющего устройства подстанции удовлетворяет условию $R < 18{,}9$ Ом. Это означает, что ЗУ обеспечивает надлежащее заземление системы, что в свою очередь помогает в выравнивании потенциала и обеспечении безопасной работы электроустановки. Важно отметить, что использование вертикальных стержней с коэффициентом использования $\eta_{\text{в.к.}} = 0{,}5$ и оптимальная длина горизонтального заземлителя позволили достичь требуемого значения сопротивления ЗУ. Это подтверждает правильность выбора параметров и конфигурации заземлителя подстанции.

7 ЭНЕРГОСБЕРЕЖЕНИЕ И РЕСУРСОСБЕРЕЖЕНИЕ ПРИ ЭКСПЛУАТАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Энергосбережение включает в себя различные меры, как правового, организационного, научного, производственного, технического, так и экономического характера, направленные на эффективное и экономное использование топливно-энергетических ресурсов, а также на использование возобновляемых источников энергии. Это важная экологическая задача, которая способствует сохранению природных ресурсов и снижению загрязнения окружающей среды, а также является экономической задачей, позволяющей снизить себестоимость товаров и услуг [10, с. 56].

Актуальность энергосбережения растет во всех странах, особенно в тех, которые не располагают богатыми энергоресурсами. Это связано с ростом цен на основные виды энергоресурсов и постепенным истощением их запасов в мире.

Энергетические ресурсы являются носителями энергии, которые используются в настоящее время или могут быть использованы в будущем. Энергетическим ресурсом может быть любой источник энергии, как естественный, так и искусственно активированный.

Ресурсосбережение представляет собой комплекс мер, направленных на бережливое и эффективное использование факторов производства, таких как капитал, земля и труд. Оно достигается через применение ресурсосберегающих и энергосберегающих технологий. Ресурсосбережение включает не только сферу производства, но и продукцию, учитывая, что продукция одной отрасли потребляется другой в рамках общественного разделения труда.

Соблюдение ресурсосбережения является важной характеристикой качества техники и технологий. Техника считается ресурсосберегающей, если она требует меньшего расхода ресурсов на своё производство и эксплуатацию.

В настоящее время ресурсосбережение является одной из приоритетных задач в экономике Республики Беларусь. Это обусловлено дефицитом многих видов ресурсов, ростом их стоимости и серьезными экологическими проблемами.

При разработке программного обеспечения необходимо учитывать основные принципы энерго- и ресурсосбережения. Экономия электроэнергии является важным аспектом современного общества, затрагивающим как производственную сферу, так и быт каждого человека. Разумное использование электроэнергии помогает избежать значительных расходов и способствует благосостоянию человека и развитию предприятия.

Существует множество способов экономии электроэнергии, некоторые из которых являются эффективными, а некоторые – менее эффективными. Для уменьшения потребляемой электроэнергии следует принимать следующие меры.

Важно использовать персональные компьютеры только в дневное время, поскольку на многих предприятиях тарифы на электроэнергию в ночное время выше, что приводит к дополнительным расходам. Необходимо обновить все компьютеры предприятия, обращая внимание на энергопотребление. Предпочтение следует отдавать стандарту *Energy Star*. Энергетический стандарт *Energy Star* – это международный стандарт энергоэффективности потребительских товаров, который позволяет снизить энергопотребление устройств на 20-30 % по сравнению с аналогами. Замена мониторов на модели с меньшим энергопотреблением также является важным шагом [11, с. 92]. При выборе нового принтера рекомендуется предпочтение отдавать струйным принтерам, которые потребляют на 80 – 90 % меньше энергии, чем лазерные. Не рекомендуется оставлять компьютеры включенными на длительное время, если они не используются. Даже в «спящем режиме» компьютер, неиспользуемый в течение двух часов, потребляет значительное количество электроэнергии. Выключение периферийных устройств, когда они не используются, также позволяет сэкономить электроэнергию [12, с. 12].

Для реализации ресурсосбережения на предприятии необходимо решить следующие задачи:

– обеспечить высокий уровень квалификации работников предприятия;

– достичь высокого интеллектуального потенциала работников;

– обеспечить технико-технологичекую независимость предприятия;

– достичь высокого уровня конкурентоспособности;

– достичь высокой эффективности менеджмента деятельности предприятия;

– обеспечить необходимый уровень экологической деятельности предприятия;

– обеспечить правовую защищённость предприятия;

– обеспечить защиту информационной безопасности предприятия.

При организации хозяйственной деятельности на предприятии подлежат реализации следующие принципы ресурсосбережения:

– принцип научной обоснованности – предполагает разумное сочетание экологических и экономических интересов общества, обеспечивающих реальные гарантии прав человека на здоровую и комфортную для жизни окружающую среду;

– принцип комплексности – предусматривает многоцелевую направленность в использовании ресурсов, развитие малоотходных и безотходных производств, глубокую переработку сырья, использование вторичных ресурсов;

– принцип экономии – основывается на рациональном использовании имеющихся экономических ресурсов с учетом их ограниченности.

Энергосбережение и ресурсосбережение являются важными аспектами

при эксплуатации программного обеспечения для управления колесными роботами и следования по заданной траектории.

Одним из ключевых аспектов является оптимизация алгоритмов движения и навигации робота. Разработчики программного обеспечения могут провести оптимизацию алгоритмов, снизив вычислительную нагрузку и использование процессорного времени. Это поможет сократить потребление электроэнергии и повысить энергоэффективность функционирования робота.

Управление энергопотреблением также играет важную роль. Программное обеспечение может предоставлять возможности управления энергопотреблением робота, например, путем автоматического перехода в режим ожидания при бездействии, выключения неиспользуемых компонентов или управления скоростью функционирования робота для оптимального использования энергии.

При планировании и следовании по траектории робота, программное обеспечение может учитывать факторы, влияющие на энергопотребление и эффективность передвижения. Расчет оптимальных траекторий может помочь сократить излишнее использование энергии и повысить общую энергоэффективность робота.

Встроенные функции программного обеспечения могут позволить мониторить и анализировать потребление ресурсов, таких как электроэнергия, процессорное время или память. Это позволяет идентифицировать узкие места и оптимизировать использование ресурсов для достижения более эффективной работы.

Регулярное обновление программного обеспечения робота также играет важную роль в энергосбережении и ресурсосбережении. Обновления могут включать исправление ошибок, оптимизацию производительности и улучшение энергоэффективности. Постоянное обновление и поддержка программного обеспечения способствуют экономии ресурсов и повышению эффективности работы робота.

В целом, энергосбережение и ресурсосбережение при эксплуатации программного обеспечения для управления колесными роботами и следования по заданной траектории являются важными задачами. Оптимизация алгоритмов, управление энергопотреблением, расчет оптимальных траекторий, мониторинг и анализ потребления ресурсов, а также регулярное обновление программного обеспечения – все это способы достижения более эффективной и устойчивого функционирования роботов при минимальном потреблении энергии и ресурсов.

ЗАКЛЮЧЕНИЕ

В ходе написания дипломной работы проведена исследовательская работа, а также систематизированы и закреплены теоретические и практические знания в рассматриваемой области. Так же разрабатывался программный комплекс для обеспечения движения аграрного колесного робота по заданной траектории, изучались существующие средства формирования спутниковых карт, а также создавались скрипты для обмена данными между клиентским приложением и *API,* устанавливаемом на роботе и скрипты, обеспечивающие движение колесного робота по заданной траектории.

При реализации ПО использовались технические средства такие, как язык программирования *Python,* для реализации системы движения аграрного колесного робота по заданной траектории, *JavaScript* для реализации клиентской части ПО, формирующей траекторию движения колесного робота на спутниковой карте.

Программные средства обеспечения движения колесного робота по заданной траектории являются важными инструментами для автоматизации планирования маршрутов колесных роботов. Они позволяют пользователям быстро и легко создавать маршруты на основе данных спутниковой карты, а также передавать их роботу для последующего выполнения.

Программный комплекс может быть использован в различных областях, таких как сельское хозяйство, логистика и транспорт.

В ходе разработки программного обеспечения проводилась верификация, включающая в себя разные типы тестирования:

– модульное тестирование;

– функциональное тестирование.

По итогам тестирования можно сделать вывод, что разработанный продукт соответствует функциональным требованиям.

Список использованных источников

1. ASUS Tinker Board: [Электронный ресурс]. – 2023. – Режим доступа: https://www.hwlibre.com/ru/asus-tinker-board/. – Дата доступа: 05.06.2023.
2. Google maps API [Электронный ресурс]. – 2023. – Режим доступа: https://developers.google.com/maps. – Дата доступа: 02.06.2023.
3. Yandex maps API [Электронный ресурс]. – 2023. – Режим доступа: https://yandex.ru/dev/maps/jsapi/doc/2.1/ref/reference/route.html. – Дата доступа: 05.06.2023.
4. Электротехника: Фильтры высоких и низких частот. /В.Х. Осадченко, Я. Ю. Волкова – ЮРАЙТ, 2022. – 81 с.
5. React: современные шаблоны для разработки приложений/ Алекс Бэнкс, Ева Порселло. – СПб: Питер, 2021. – 320 с.
6. Чистый Python. Тонкости программирования для профи / Дэн Бейдер. – СПб: Питер, 2020. – 288 с.
7. Python к вершинам мастерства. / Л.Рамальо – ДМК, 2022. – 898 с.
8. Python. Книга рецептов. / Д. Бизли, К. Джонс – СПб: Питер, 2019. – 646 с.
9. Расчет заземления [Электронный ресурс]. – 2023. – Режим доступа: https://www.uss-electro.ru/questions/raschet-zazemln/. – Дата доступа: 06.06.2023.
10. Андрижиевский, А. А. Энергосбережение и энергетический менеджмент : учеб. пособие для студ. / А. А. Андрижиевский, В. И. Володин. – 2-е изд., испр. – Мн. : Вышэйшая школа, 2017. – 294 с.
11. Организация энергосбережения (энергоменеджмент). Решения ЗСМК-НКМКНТМК-ЕВРАЗ : учебное пособие / под ред. В. В. Кондратьева – М.: ИНФРАМ, 2017. – 108 с.
12. Ляшенко, Д. Д. Ресурсосбережение: инновации в энергосбережении / Д. Д. Ляшенко, В. И. Мартынович // Международное научное обозрение проблем и перспектив современной науки и образования. Сборник статей по материалам XXX Международной научно-практической конференции, 2018. – С. 32.

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг программы

Листинг pythonApi.py

```
import json
import os
from flask import Flask
from flask_cors import CORS
from flask_socketio import SocketIO
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler

# CONSTANTS
global pointsToMove
pointsToMove = []
filename = 'markers.json'
dir_path = os.path.dirname(os.path.abspath(__file__))
file_path = os.path.join(dir_path, filename)

API_URI = 'http://localhost:5000'

app = Flask(__name__)
CORS(app)
socketio = SocketIO(app, cors_allowed_origins='*')

#класс проверяющий состояние файла с координатами
class FileModifiedEventHandler(FileSystemEventHandler):
    def on_modified(self, event):
        if event.src_path == file_path:
            print(f"{filename} has been modified, reloading data...")
            data = read_data()
            if data:
                parseData(data)

#функция просмотра файла
def watch_file():
    event_handler = FileModifiedEventHandler()
    observer = Observer()
    observer.schedule(event_handler, dir_path, recursive=False)
    observer.start()
    print(f"Watching for changes to {filename}...")


#класс описывающий координату
class Coords:
    def __init__(self, id, lat, lng):
        self.id = id
        self.lat = lat
        self.lng = lng

    def __repr__(self):
        return f"Coords(id={self.id}, lat={self.lat}, lng={self.lng})"

@socketio.on('connect')
def connect():
```

```
    print('Client connected')


@socketio.on('disconnect')
def disconnect():
    print('Client disconnected')


@socketio.on('data')
def handle_data(json_str):
    if not json_str:
        socketio.emit('error', {'message': 'invalid data'})
        return

    data = json.loads(json_str)
    socketio.emit('response', {'message': 'Server received data from client'})
    parseData(data)

#обработка данных пришедших с сервера
def parseData(data):
    points = []
    if not isinstance(data, list):
        print('Error: Input data is not a list')
        return []

    for item in data:
        id = item['_id']
        lat = item['cords']['lat']
        lng = item['cords']['lang']
        point = Coords(id, lat, lng)
        points.append(point)
    global pointsToMove
    pointsToMove = points
    print(pointsToMove)

#функция чтения данных из файла
def read_data():
    if os.path.exists(filename):
        with open(filename, 'r') as f:
            data = json.load(f)
            return data
    else:
        print(f'Error: file {filename} does not exist')
        return []

#запуск скрипта
if __name__ == '__main__':
    watch_file()
    socketio.run(app, host='0.0.0.0', port=5000)
```

Листинг RobotController.py

```
import time
import ASUS.GPIO as GPIO
from gps import gps, WATCH_ENABLE

#класс управления роботом
class RobotController:
    def __init__(self):
        GPIO.setmode(GPIO.ASUS)
```

```
        self.pwm_pin1 = 238
        self.pwm_pin2 = 239
        self.relay_pin1 = 10
        self.relay_pin2 = 11
        self.gpsd = gps(mode=WATCH_ENABLE)
        self.pwm1 = None
        self.pwm2 = None

#настройка портов и пинов
    def setup(self):
        GPIO.setup(self.pwm_pin1, GPIO.OUT)
        GPIO.setup(self.pwm_pin2, GPIO.OUT)
        GPIO.setup(self.relay_pin1, GPIO.IN)
        GPIO.setup(self.relay_pin2, GPIO.IN)
        self.pwm1 = GPIO.PWM(self.pwm_pin1, 100)
        self.pwm2 = GPIO.PWM(self.pwm_pin2, 100)
        self.pwm1.start(0)
        self.pwm2.start(0)

#установка скорости
    def set_speed(self, speed1, speed2):
        self.pwm1.ChangeDutyCycle(speed1)
        self.pwm2.ChangeDutyCycle(speed2)

#остановка мотора
    def stop_motors(self):
        self.set_speed(0, 0)


#поворот на лево
def turn_left(self, speed):
        self.set_speed(0, speed)

#поворот на право
    def turn_right(self, speed):
        self.set_speed(speed, 0)

#получение текущей координаты
    def get_gps_coordinates(self):
        self.gpsd.next()

        if self.gpsd.valid and self.gpsd.fix.latitude and self.gpsd.fix.longitude:
            latitude = self.gpsd.fix.latitude
            longitude = self.gpsd.fix.longitude
            return latitude, longitude
        else:
            return None, None

#функция движения к координате
    def move_to_coordinate(self, target_latitude, target_longitude):
        current_latitude, current_longitude = self.get_gps_coordinates()

        if current_latitude is None or current_longitude is None:
            return

        if target_latitude > current_latitude:
            self.set_speed(100, 100)  # Move forward
        elif target_longitude < current_longitude:
            self.turn_left(100)  # Turn left
```

```
        elif target_longitude > current_longitude:
            self.turn_right(100)  # Turn right
        else:
            self.stop_motors()  # Stop if already at the target coordinate

#функция движения к целевым точкам
    def move_to_target_points(self, target_points):
        for point in target_points:
            target_latitude, target_longitude = point
            self.move_to_coordinate(target_latitude, target_longitude)
            time.sleep(1)  # Adjust the sleep time between points as desired

#запуск движения
    def start(self):
        self.setup()

        try:
            while True:
                relay_value1 = GPIO.input(self.relay_pin1)
                relay_value2 = GPIO.input(self.relay_pin2)

                if relay_value1 == 0 and relay_value2 == 0:
                    self.stop_motors()
                elif relay_value1 == 1 and relay_value2 == 0:
                    self.turn_left(100)
                elif relay_value1 == 0 and relay_value2 == 1:
                    self.turn_right(100)
                elif relay_value1 == 1 and relay_value2 == 1:
                    self.set_speed(100, 100)
                else:
                    self.stop_motors()

                target_points = [(37.123, -122.456), (37.456, -122.789), (37.789, -122.123)]
                self.move_to_target_points(target_points)

                time.sleep(1)

        except KeyboardInterrupt:
            self.stop_motors()

        except Exception as e:
            print("Произошла ошибка:", str(e))
            self.stop_motors()

        finally:
            GPIO.cleanup()

# Instantiate the RobotController class and start the robot
robot = RobotController()
robot.start()
```

Листинг клиентского приложения

```
import io from "socket.io-client";
import { API_URI } from "../components/constants";

const socket = io.connect(API_URI, {
  transports: ["websocket", "polling"],
});
```

//отправка данных на сервер

```
const sendDataToApi = (data) => {
  const sendData = () => {
    if (Array.isArray(data) && data.length > 0) {
      socket.emit("data", JSON.stringify(data));
    } else {
      console.error("Error: Invalid data format");
    }
  };

  const eventHandlers = {
    data: (data) => {
      console.log("Received data from server:", data);
    },
    response: (data) => {
      console.log("Response from server:", data.message);
    },
    connect_error: (error) => {
      console.error("Error connecting to server:", error.message);
    },
    error: (error) => {
      console.error("Socket error:", error.message);
    },
  };

  Object.keys(eventHandlers).forEach((eventName) => {
    socket.on(eventName, eventHandlers[eventName]);
  });

  return { sendData };
};

export default sendDataToApi;
```

//константы

```
export const API_KEY = "736ec73a-b570-41dd-8444-259a9486cf59";
export const ROUTE_MODE = "pedestrian";
export const MAX_MARKER_NUMBERS = "7";
export const CURRENT_MAP_STATE = {
  center: [52.40621344277536, 30.938249317326758],
  zoom: 17,
};

export const API_URI = "http://localhost:5000";
```

//компонент меню

```
import React from "react";
import  GetScriptFromMapButton  from "../ButtonsComponents/GetScriptFromMapButton";
import  ClearPointsButton  from "../ButtonsComponents/ClearPointsButton";
import  ClearLastPointButton  from "../ButtonsComponents/ClearLastPointButton";
import  SendDataToApiButton  from "../ButtonsComponents/SendDataToApiButton";

function RightSideMenu(props) {
    return (
    <div className="right_menu">
      <div className="buttons-container">
        <GetScriptFromMapButton onClick={props.buttonHandler.getScriptFromMap} />
        <ClearPointsButton onClick={props.buttonHandler.clearRoutes} />
        <ClearLastPointButton onClick={props.buttonHandler.clearLastPoint} />
```

```
        <SendDataToApiButton onClick={props.buttonHandler.sendDataToApi} />
      </div>
    </div>
  );
}
export default RightSideMenu;
```

//компонент карты

```
import React from "react";
import { useState, useRef } from "react";
import {
  YMaps,
  Map,
  SearchControl,
  GeolocationControl,
  TypeSelector,
} from "react-yandex-maps";
import {
  API_KEY,
  ROUTE_MODE,
  MAX_MARKER_NUMBERS,
  CURRENT_MAP_STATE,
} from "../constants";
import RightSideMenu from "../MenuComponents/RightSideMenu";
import SendDataToApi from "../../services/sendDataToApi";

function MapComponent() {
  const [ymaps, setYmaps] = useState(null);
  const [markers, setMarkers] = useState([]);

  const savedMarkers = markers.map((marker) => {
    let obj = {
      _id: marker.id,
      cords: { lat: marker.coords[0], lang: marker.coords[1] },
    };
    return obj;
  });

  const { sendData } = SendDataToApi(savedMarkers);

  let multiRoute = useRef(null);

  const addPointToMap = (event) => {
    const coords = event.get("coords");
    if (markers.length < MAX_MARKER_NUMBERS) {
      const newMarker = {
        id: markers.length + 1,
        coords,
        hintContent: `Точка ${markers.length + 1}`,
      };
      setMarkers([...markers, newMarker]);
    }
  };

  const getRoute = (ref) => {
    if (ymaps) {
      if (!multiRoute.current) {
        // Создаем новый маршрут при первом вызове функции
        multiRoute.current = new ymaps.multiRouter.MultiRoute(
```

```
        {
          referencePoints: markers.map((marker) => marker.coords),
          params: {
            results: 1,
            routingMode: ROUTE_MODE,
          },
        },
        {
          routeActiveStrokeWidth: 6,
          routeActiveStrokeStyle: "solid",
          routeActiveStrokeColor: "#000",
          routeActiveMarkerEnabled: true,
          routeActiveMarkerType: "start",
          routeActiveMarkerVisible: true,
          routeMarkerIconLayout: "default#image",
          routeMarkerIconImageHref: "path/to/image.png",
          routeMarkerIconImageSize: [32, 32],
          routeMarkerIconImageOffset: [-16, -16],
          showIntermediatePoints: false,
          boundsAutoApply: true,
          zoomMargin: 0,
          useMapMargin: true,
          mapStateAutoApply: true,
          layerKey: "layer#map",
          zIndex: 100,
          balloonAutoPan: true,
          balloonContentLayoutWidth: 200,
          minObstacleHeight: 0,
          maxObstacleHeight: 50,
          wayPointStartIconColor: "#FFFFFF",
          wayPointStartIconFillColor: "#B3B3B3",
          wayPointFinishIconColor: "#FFFFFF",
          wayPointFinishIconFillColor: "#B3B3B3",
        }
      );
      ref.geoObjects.add(multiRoute.current);
    } else {
      multiRoute.current.model.setReferencePoints(
        markers.map((marker) => marker.coords)
      );
    }
  }
};

const buttonHandler = {
  getScriptFromMap: () => {
    if (!ymaps) return;
    const data = savedMarkers;
    const stateString = JSON.stringify(data);
    const blob = new Blob([stateString], { type: "application/json" });
    const url = URL.createObjectURL(blob);
    const a = document.createElement("a");
    a.download = "markers.json";
    a.href = url;
    a.click();
  },
  clearRoutes: () => {
    setMarkers([]);
    if (multiRoute.current) {
```

```
      multiRoute.current.model.setReferencePoints([]);
    }
  },
  clearLastPoint: () => {
    if (markers.length > 0) {
      setMarkers(markers.slice(0, markers.length - 1));
    }
  },
  sendDataToApi: () => {
    if (!ymaps) return;
    sendData();
  },
};

 return (
  <div className="map_container">
   <div className="map_wrapper-content">
    <YMaps
     query={{
      apikey: API_KEY,
      lang: "ru_RU",
     }}
    >
     <Map
      width="70%"
      height="80vh"
      modules={["multiRouter.MultiRoute"]}
      onLoad={(ymaps) => setYmaps(ymaps)}
      state={CURRENT_MAP_STATE}
      instanceRef={(ref) => ref && getRoute(ref)}
      onClick={addPointToMap}
     >
      <GeolocationControl options={{ float: "left" }} />
      <SearchControl options={{ float: "right" }} />
      <TypeSelector options={{ float: "right" }} />
     </Map>
    </YMaps>
    <RightSideMenu buttonHandler={buttonHandler} />
   </div>
  </div>
 );
}
export default MapComponent;
```

//обработка кнопок

```
import React from 'react';

function ClearLastPointButton(props) {
 return (
  <button className='menu-button clearLastPointButton' onClick={props.onClick}>Очистить последнюю
добавленную точку</button>
 );
}

export default ClearLastPointButton;

import React from 'react';
```

```
function ClearPointsButton(props) {
  return (
    <button className='menu-button clearPointsButton' onClick={props.onClick}>Очистить все точки</button>
  );
}

export default ClearPointsButton;

import React from 'react';

function GetScriptFromMapButton(props) {
  return (
    <button className='menu-button GetScriptFromMapButton' onClick={props.onClick}>Сгенерировать скрипт маршрута</button>
  );
}

export default GetScriptFromMapButton;

import React from 'react';

function SendDataToApiButton(props) {
  return (
    <button className='menu-button SendDataToApiButton' onClick={props.onClick}>Отправить данные на api</button>
  );
}
export default SendDataToApiButton;

import React from "react";
import  MapComponent  from "../MapsComponents/MapComponent.js";
 function ContentWrapper() {
  return (
    <div className="content_wrapper">
      <MapComponent />
    </div>
  );
}

export default ContentWrapper;

import React from 'react';

function Footer() {
  return (
    <footer className="footer">
      <div className="footer-item">
        <span>Расшивалов Никита Игоревич ИТП-41</span>
        <span>ГГТУ им. П.О. Сухого</span>
      </div>
    </footer>
  );
}

export default Footer;

import React from "react";
import  Logo  from "../../content/logo.png";
```

```
function Header() {
  return (
    <header className="header">
      <div className="header-item">
        <img src={Logo} alt="logo"></img>
        <div className="header-title">
          <span className="title-text">Построение маршрутра движения колесного робота</span>
        </div>
      </div>
    </header>
  );
}

export default Header;

import React from "react";
import Header from "./Header";
import Footer from "./Footer";
import ContentWrapper from "./ContentWrapper";

export function Layout() {
  return (
    <>
      <Header />
      <ContentWrapper />
      <Footer />
    </>
  );
}

export default Layout;

import "./App.css";
import { Layout } from './components/shared/Layout';

function App() {
  return (
    <Layout/>
  );
}

export default App;

import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <App />
);

//стили
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
```

```
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}

@import url("https://fonts.googleapis.com/css2?family=Inter&display=swap");

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

html,
body {
  height: 100%;
  width: 100%;
  position: absolute;
  background-color: #e7e7e7;
  font-family: "Inter";
  font-weight: 900;
}

#root {
  position: relative;
  display: flex;
  flex-direction: column;
  min-height: 100vh;
  justify-content: center;
}
.header {
  background-color: #8a76e3;
  display: flex;
  flex-direction: column;
  opacity: 1;
}

.header-item {
  display: flex;
  flex-direction: row;
  align-items: center;
  height: 12vh;
}

.header-item img {
  width: 7vw;
  height: 10vh;
  opacity: 0.7;
  margin-right: 110px;
  margin-left: 20px;
}

.header-title {
```

```
  display: flex;
  align-items: center;
  justify-content: center;
  text-align: center;
}

.title-text {
  font-size: 5vh;
  opacity: 0.8;
}

.footer {
  flex-shrink: 0;
  height: 6vh;
  font-size: 25px;
  opacity: 0.8;
  background: #8a76e3;
}

.footer-item {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
}
.footer-item:first-child {
  margin-left: 5px;
}
.footer-item:last-child {
  margin-right: 5px;
}

.content_wrapper {
  flex-grow: 1;
  background: #c5b9ec;
  height: 80%;
}

/* wrapper */
.map_container {
  height: 100%;
}

.map_wrapper-content {
  display: flex;
  flex-direction: row;
  padding: 5px;
}
.right_menu {
  width: 30%;
}

/* bottons */
.buttons-container {
  display: flex;
  flex-direction: column;
}

.menu-button {
  position: relative;
```

```
  background: #8a76e3;
  color: #000;
  width: 95%;
  height: 10vh;
  border: 0;
  font-size: 18px;
  border-radius: 4px;
  transition: 0.6s;
  overflow: hidden;
  border-radius: 10% / 100%;
  margin: 5px;
}
.menu-button:focus {
  outline: 0;
}
.menu-button:before {
  content: "";
  display: block;
  position: absolute;
  background: rgba(255, 255, 255, 0.5);
  width: 60px;
  height: 100%;
  left: 0;
  top: 0;
  opacity: 0.5;
  filter: blur(30px);
  transform: translateX(-100px) skewX(-15deg);
}
.menu-button:after {
  content: "";
  display: block;
  position: absolute;
  background: rgba(255, 255, 255, 0.2);
  width: 30px;
  height: 100%;
  left: 30px;
  top: 0;
  opacity: 0;
  filter: blur(5px);
  transform: translateX(-100px) skewX(-15deg);
}
.menu-button:hover {
  background: #b94bd2;
  cursor: pointer;
}
.menu-button:hover:before {
  transform: translateX(300px) skewX(-15deg);
  opacity: 0.6;
  transition: 0.7s;
}
.menu-button:hover:after {
  transform: translateX(300px) skewX(-15deg);
  opacity: 1;
  transition: 0.7s;
}
```

ПРИЛОЖЕНИЕ Б
(обязательное)
Руководство системного программиста

1. Общие сведения о программных средствах обеспечения движения колесного робота по заданной траектории.

Разработанный продукт предназначен управления движение колесного робота по заданной траектории. Управление системой выполняется с помощью браузера на ПК или через *VNC client*.

Для корректной работы ПО необходимо соблюдение следующих требований:

– версия *Debian* от 9.0 для работы ПО на одноплатном компьютере;

– 2 Гб оперативной памяти;

– наличие *VNC* сервера на одноплатном компьютере.

2. Структура программных средств обеспечения движения колесного робота по заданной траектории.

Структурно приложение разделено на два компонента:

– клиентское приложение, формирования маршрута на спутниковой карте;

– *API*, устанавливаемое на одноплатный компьютер для управления движением по заданной траектории.

3. Настройка программных средств обеспечения движения колесного робота по заданной траектории.

Для настройки клиентского приложения необходимо в файле *constants.js* указать адрес сокета, по которому оно будет связано с серверной частью приложения.

Для настройки сервера необходимо, в файле *pythonApi.py* указать адрес сокета, по которому он будет связан с клиентским приложением.

Для возможности удаленного подключения к одноплатному компьютеру и внесения правок на установленный софте, на него необходимо установить любой *VNC server*, а на любой ПК любой *VNC client*.

4. Проверка программных средств обеспечения движения колесного робота по заданной траектории.

Успешный запуск движения колесного робота по заданной траектории свидетельствует о корректной работе программных средств.

5. Дополнительные возможности.

Дополнительные возможности в приложении отсутствуют.

6. Сообщения системному администратору.

При возникновении ошибок работы серверной части необходимо убедиться, что указанный адрес сокета не занят другим приложением.

ПРИЛОЖЕНИЕ В
(обязательное)
Руководство программиста

1. Назначение и условия применения программных средств обеспечения движения колесного робота по заданной траектории.

Разработанный продукт предназначен для автоматизации движения колесного робота по заданной траектории. Управление системой выполняется с помощью веб-приложения.

Для корректной работы ПО необходимо соблюдение следующих требований:

– версия *Debian* от 9.0 для работы серверной части приложения;

– 2 Гб оперативной памяти;

– наличие *VNC* сервера на одноплатном компьютере.

2. Характеристики программных средств обеспечения движения колесного робота по заданной траектории.

Управление системой выполняется с помощью веб-приложения.

3. Обращение к программным средствам обеспечения движения колесного робота по заданной траектории.

Для просмотра и редактирования исходного кода клиентской части приложения необходимо открыть папку *route-definition-app* с помощью среды разработки, поддерживающей разработку *на React-js*, например, *Visual Studio Code*. Для работы с кодом серверной части приложения необходима среда разработки, позволяющая работу с *Python*, например *Visual Studio Code* или *Geditor*(на одноплатном компьютере) .

4. Входные и выходные данные.

Входными данными является скрипт, содержащий маршрут движения колесного робота.

Выходными данными является информация, отображаемая о сервере в клиентском приложении.

5. Сообщения.

При возникновении ошибок работы какого-либо из приложений необходимо проверить информацию, выводимую в консоль для принятия дальнейших действий по решению проблемы.

ПРИЛОЖЕНИЕ Г
(обязательное)
Руководство пользователя

1. Введение.

Руководство пользователя информирует пользователя базовыми знаниями по эксплуатации разработанного продукта.

Разработанный продукт предназначен для обеспечения движения колесного робота по заданной траектории.

Для работы с программным обеспечением пользователь должен иметь начальные сведения и навыки работы с веб-браузером.

2. Назначение и условия применения.

Разработанный продукт предназначен обеспечения движения колесного робота по заданной траектории. Маршрут следования формируется на клиентской части приложения. Управление системой выполняется с помощью веб-приложения.

Условия необходимые для функционирования веб-приложения:

– наличие подключения к сети интернет либо локальной сети в зависимости от настройки программного комплекса;

– наличие любого современного веб-браузера.

3. Подготовка к работе.

Для работы необходимо открыть веб-приложение.

4. Описание операций.

Операция 1: Добавление точек на спутниковую карту;

Условия необходимые для выполнения: запущенное веб-приложение;

Подготовительные действия: необходимо предварительно добавленное соединение;

Основные действия: на карте кликнуть по тем точкам, которые должны присутствовать в траектории движения робота.

Заключительные действия: отсутствуют.

Ресурсы, расходуемые на операцию: время на добавление точек маршрута.

Операция 2: Удаление последней добавленной точки;

Условия необходимые для выполнения: запущенное веб-приложение;

Подготовительные действия: необходимо предварительно добавленное соединение;

Основные действия: нажать на кнопку удаления последней добавленной точки на веб-странице.

Операция 3: Очистка маршрута;

Условия необходимые для выполнения: запущенное веб-приложение;

Подготовительные действия: необходимо предварительно добавленное соединение;

Основные действия: нажать на кнопку очистки всех точек маршрута на веб-странице.

Заключительные действия: отсутствуют.

Операция 4: Генерация скрипта маршрута;

Условия необходимые для выполнения: запущенное приложение;

Подготовительные действия: необходимо предварительно добавленное соединение;

Основные действия: нажать на кнопку генерации маршрута на веб-странице «Сгенерировать скрипт».

Заключительные действия: отсутствуют.

Ресурсы, расходуемые на операцию: время генерацию скрипта.

Операция 5: Отправка скрипта с маршрутом на сервер;

Условия необходимые для выполнения: запущенное приложение и запущенный сервер;

Подготовительные действия: необходимо предварительно добавленное соединение;

Основные действия: нажать на кнопку отправки маршрута на сервер «Отправить данные на *api*».

Заключительные действия: отсутствуют.

Ресурсы, расходуемые на операцию: время на отправку скрипта на сервер.

5. Аварийные ситуации.

Для избежания возникновения аварийных ситуаций необходимо использовать актуальную версию *Debian*.

В случае отказа работы приложения рекомендуется перезапустить приложение либо проверить наличие подключения к сети.

6. Рекомендации по освоению.

Для освоения в приложении необходимо изучить информацию ознакомиться с руководством пользователя.

ПРИЛОЖЕНИЕ Д
(справочное)
Формулы расчета экономической эффективности

Техническая прогрессивность программного продукта определяется коэффициентом эквивалентности и рассчитывается по формуле (Д.1):

$$K_{\text{эк}} = \frac{K_{\text{т.н}}}{K_{\text{т.б}}}, \qquad \text{(Д.1)}$$

где $K_{\text{т.н}}$, $K_{\text{т.б}}$ – коэффициенты технического уровня нового и базисного программного продукта, которые можно рассчитать по формуле (Д.2):

$$K_{\text{т}} = \sum_{i=1}^{n} \beta \frac{P_i}{P_{\text{э}}}, \qquad \text{(Д.2)}$$

где β – коэффициенты весомости *i*-го технического параметра;

n – число параметров;

P_i – численное значение *i*-го технического параметра сравниваемого программного продукта;

$P_{\text{э}}$ – численное значение *i*-го технического параметра эталона.

Коэффициент изменения функциональных возможностей ($K_{\text{ф.в}}$) нового программного обеспечения рассчитывается по формуле (Д.3):

$$K_{\text{ф.в}} = \frac{K_{\text{ф.в.н}}}{K_{\text{ф.в.б}}}, \qquad \text{(Д.3)}$$

где $K_{\text{ф.в.н}}$, $K_{\text{ф.в.б}}$ – балльная оценка неизмеримых показателей нового и базового изделия соответственно.

Конкурентоспособность нового программного продукта рассчитывается по формуле (Д.4):

$$K_{\text{и}} = \frac{K_{\text{эк}} \cdot K_{\text{ф.в}} \cdot K_{\text{н}}}{K_{\text{ц}}}, \qquad \text{(Д.4)}$$

где $K_{\text{н}}$ – коэффициент соответствия нового программного продукта нормативам ($K_{\text{н}} = 1$);

$K_{\text{ц}}$ – коэффициент цены потребления.

Общий объем разрабатываемого программного обеспечения ($V_{\text{о}}$) вычисляется по формуле (Д.5):

$$V_{\text{о}} = \sum_{i=1}^{n} V_i , \qquad (\text{Д.5})$$

где V_i – объем отдельной функции ПО;

n – общее число функций.

Уточненный объем программного обеспечения ($V_{\text{у}}$) определяется по формуле (Д.6):

$$V_{\text{у}} = \sum_{i=1}^{n} V_{\text{у}i} , \qquad (\text{Д.6})$$

где $V_{\text{у}i}$ – уточненный объем отдельной функции ПО в строках исходного кода.

Коэффициент повышения сложности ПО ($K_{\text{с}}$) рассчитывается по формуле (Д.7):

$$K_{\text{с}} = 1 + \sum_{i=1}^{n} K_i , \qquad (\text{Д.7})$$

где K_i – коэффициент, соответствующий степени повышения сложности;

n – количество учитываемых характеристик.

Нормативная трудоемкость ПО для стадии ТЗ вычисляется по формуле (Д.8), для стадии ЭП по формуле (Д.9), для стадии ТП по формуле (Д.10), для стадии РП по формуле (Д.11), для стадии ВН по формуле (Д.12):

$$\text{Т}_{\text{у.т.з}} = \text{Т}_{\text{н}} \cdot K_{\text{т.з}} \cdot K_{\text{с}} \cdot K_{\text{н}} \cdot K_{\text{у.р}}, \qquad (\text{Д.8})$$

$$\text{Т}_{\text{у.э.п}} = \text{Т}_{\text{н}} \cdot K_{\text{э.п}} \cdot K_{\text{с}} \cdot K_{\text{н}} \cdot K_{\text{у.р}}, \qquad (\text{Д.9})$$

$$\text{Т}_{\text{у.т.п}} = \text{Т}_{\text{н}} \cdot K_{\text{т.п}} \cdot K_{\text{с}} \cdot K_{\text{н}} \cdot K_{\text{у.р}}, \qquad (\text{Д.10})$$

$$\text{Т}_{\text{у.р.п}} = \text{Т}_{\text{н}} \cdot K_{\text{р.п}} \cdot K_{\text{с}} \cdot K_{\text{н}} \cdot K_{\text{т}} \cdot K_{\text{у.р}}, \qquad (\text{Д.11})$$

$$Т_{у.в.н} = Т_{н} \cdot K_{в.н} \cdot K_{с} \cdot K_{н} \cdot K_{у.р}, \qquad (Д.12)$$

где $K_{т.з}$, $K_{э.п}$, $K_{т.п}$, $K_{р.п}$ и $K_{в.н}$ – значения коэффициентов удельных весов трудоемкости стадий разработки ПО в общей трудоемкости ПО.

Общая трудоемкость разработки программного обеспечения ($Т_{о}$) вычисляется по формуле (Д.13):

$$Т_{о} = \sum_{i=1}^{n} Т_{уi}, \qquad (Д.13)$$

где $Т_{уi}$ – нормативная (скорректированная) трудоемкость разработки ПО на i-й стадии, чел.-дн.;

n – количество стадий разработки.

Суммарные затраты на разработку программного обеспечения ($З_{р}$) определяются по формуле (Д.14):

$$З_{р} = З_{тр} + З_{эт} + З_{тех} + З_{м.в} + З_{мат} + З_{общ.пр} + З_{непр}. \qquad (Д.14)$$

Расходы на оплату труда разработчиков с отчислениями ($З_{тр}$) определяются по формуле (Д.15):

$$З_{тр} = ЗП_{осн} + ЗП_{доп} + ОТЧ_{зп}, \qquad (Д.15)$$

где $ЗП_{осн}$ – основная заработная плата разработчиков, руб.;

$ЗП_{доп}$ – дополнительная заработная плата разработчиков, руб.;

$ОТЧ_{зп}$ – сумма отчислений от заработной платы (социальные нужды, страхование от несчастных случаев), руб.

Основная заработная плата вычисляется по формуле (Д.16):

$$ЗП_{осн} = С_{ср.час} \cdot Т_{о} \cdot K_{ув}, \qquad (Д.16)$$

где $С_{ср.час}$ – средняя часовая тарифная ставка, руб./час;

$Т_{о}$ – общая трудоемкость разработки, чел.-час;

$K_{ув}$ – коэффициент доплаты стимулирующего характера, $K_{ув}$ = 1,8.

Средняя часовая тарифная ставка определяется по формуле (Д.17):

$$С_{ср.час} = \frac{\sum_i С_{чi} \cdot n_i}{\sum_i n_i}, \qquad (Д.17)$$

где $С_{чi}$ – часовая тарифная ставка разработчика i-й категории, руб./час;

n_i – количество разработчиков i-й категории.

Часовая тарифная ставка вычисляется по формуле (Д.18):

$$С_ч = \frac{С_{м1} \cdot Т_{к1}}{F_{мес}}, \qquad (Д.18)$$

где $С_{м1}$ – тарифная ставка первого разряда;

$Т_{к1}$ – тарифный коэффициент.

Дополнительная заработная плата рассчитывается по формуле (Д.19):

$$ЗП_{доп} = \frac{ЗП_{осн} \cdot Н_{доп}}{100}, \qquad (Д.19)$$

где $Н_{доп}$ – норматив на дополнительную заработную плату разработчиков.

Отчисления от основной и дополнительной заработной платы рассчитываются по формуле (Д.20):

$$ОТЧ_{с.н} = \frac{\left(ЗП_{осн} + ЗП_{доп}\right) \cdot Н_{з.п}}{100}, \qquad (Д.20)$$

где $Н_{з.п}$ – процент отчислений на социальные нужды и обязательное страхование от суммы основной и дополнительной заработной платы ($Н_{з.п}$ = 34 процента).

Затраты машинного времени вычисляются по формуле (Д.21):

$$З_{м.в} = С_ч \cdot K_т \cdot t_{эвм}, \qquad (Д.21)$$

где $С_ч$ – стоимость одного часа машинного времени, руб./ч;

$K_т$ – коэффициент мультипрограммности, показывающий распределение времени работы ЭВМ в зависимости от количества пользователей ЭВМ, $K_т = 1$;

$t_{эвм}$ – машинное время ЭВМ, необходимое для разработки и отладки проекта, ч.

Стоимость одного часа машинного времени рассчитывается с использованием формулы (Д.22):

$$С_{ч} = \frac{ЗП_{об} + З_{ар} + З_{ам} + З_{э.п} + З_{в.м} + З_{т.р} + З_{пр}}{F_{эвм}}, \qquad (Д.22)$$

где $ЗП_{об}$ – затраты на заработную плату обслуживающего персонала с учетом всех отчислений, руб./год;

$З_{ар}$ – стоимость аренды помещения под размещение вычислительной техники, руб./год;

$З_{ам}$ – амортизационные отчисления за год, руб./год;

$З_{э.п}$ – затраты на электроэнергию, руб./год;

$З_{в.м}$ – затраты на материалы, необходимые для обеспечения нормальной работы ПЭВМ (вспомогательные), руб./год;

$З_{т.р}$ – затраты на текущий и профилактический ремонт ЭВМ, руб./год;

$З_{пр}$ – прочие затраты, связанные с эксплуатацией ПЭВМ, руб./год;

$F_{эвм}$ – действительный фонд времени работы ЭВМ, час/год.

Затраты на заработную плату обслуживающего персонала вычисляются по формуле (Д.23):

$$ЗП_{об} = \frac{ЗП_{осн.об} + ЗП_{доп.об} + ОТЧ_{эп.об}}{100}, \qquad (Д.23)$$

Основная заработная плата обслуживающего персонала вычисляется по формуле (Д.24):

$$ЗП_{осн.об} = 12 \cdot \sum_{i} \left(С_{м.об i} \cdot n_i \right), \qquad (Д.24)$$

Дополнительная заработная плата обслуживающего персонала вычисляется по формуле (Д.25):

$$ЗП_{доп.об} = \frac{ЗП_{осн.об} \cdot Н_{доп}}{100}, \qquad (Д.25)$$

Сумма отчислений от заработной платы вычисляется по формуле (Д.26):

$$\text{ОТЧ}_{\text{зп.об}} = \frac{\left(\text{ЗП}_{\text{осн.об}} + \text{ЗП}_{\text{доп.об}}\right) \cdot \text{Н}_{\text{зп}}}{100}, \qquad \text{(Д.26)}$$

где $\text{ЗП}_{\text{осн.об}}$ – основная заработная плата обслуживающего персонала, руб.;

$\text{ЗП}_{\text{доп.об}}$ – дополнительная заработная плата обслуживающего персонала, руб.;

$\text{ОТЧ}_{\text{зп.об}}$ – сумма отчислений от заработной платы (социальные нужды, страхование от несчастных случаев), руб.;

$Q_{\text{эвм}}$ – количество обслуживаемых ПЭВМ, шт.;

$\text{С}_{\text{м.об}i}$ – месячная тарифная ставка i-го работника, руб.;

n – численность обслуживающего персонала, чел.;

$\text{Н}_{\text{доп}}$ – процент дополнительной заработной платы обслуживающего персонала от основной;

$\text{Н}_{\text{зп}}$ – процент отчислений на социальные нужды и обязательное страхование от суммы основной и дополнительной заработной платы.

Годовые затраты на аренду помещения вычисляются по формуле (Д.27):

$$\text{З}_{\text{ар}} = \frac{\text{С}_{\text{ар}} \cdot S}{Q_{\text{эвм}}}, \qquad \text{(Д.27)}$$

где $\text{С}_{\text{ар}}$ – средняя годовая ставка арендных платежей, руб./м^2;

S – площадь помещения, м^2.

Сумма годовых амортизационных отчислений ($\text{З}_{\text{ам}}$) определяется по формуле (Д.28):

$$\text{З}_{\text{ам}} = \frac{\sum_i \text{З}_{\text{пр}i} \cdot \left(1 + K_{\text{доп}}\right) \cdot m_i \cdot \text{Н}_{\text{ам}i}}{Q_{\text{эвм}}}, \qquad \text{(Д.28)}$$

где $\text{З}_{\text{пр}i}$ – затраты на приобретение i-го вида основных фондов, руб;

$K_{\text{доп}}$ – коэффициент, дополнительных затраты, связанные с доставкой, монтажом и наладкой оборудования, $K_{\text{доп}}$ = 13 процентов от $\text{З}_{\text{пр}}$;

$\text{З}_{\text{пр}i}\left(1 + K_{\text{доп}}\right)$ – балансовая стоимость ЭВМ, руб;

$\text{Н}_{\text{ам}i}$ – норма амортизации, процентов.

Стоимость электроэнергии, потребляемой за год, вычисляется с использованием формулы (Д.29):

$$З_{эвм} = \frac{М_{сум} \cdot F_{эвм} \cdot С_{эл} \cdot A}{Q_{эвм}}, \qquad (Д.29)$$

где $М_{сум}$ – паспортная мощность ПЭВМ, кВт;

$М_{сум}$ = 0,41 кВт;

$С_{эл}$ – стоимость одного кВт-часа электроэнергии, руб;

А – коэффициент интенсивного использования мощности, А=0,98…0,9.

Годовой фонд времени работы ПЭВМ рассчитывается по формуле (Д.30):

$$F_{эвм} = \left(Д_{г} - Д_{вых} - Д_{пр}\right) \cdot F_{см} \cdot K_{см} \cdot \left(1 - K_{пот}\right), \qquad (Д.30)$$

где $Д_{г}$ – общее количество дней в году, $Д_{г}$ = 365 дней;

$Д_{вых}$, $Д_{пр}$ – число выходных и праздничных дней в году, $Д_{вых} + Д_{пр}$ = 121 дней;

$F_{см}$ – продолжительность 1 смены, $F_{см}$ = 8 часов;

$K_{см}$ – коэффициент сменности, $K_{см}$= 1;

$K_{пот}$ – коэффициент, учитывающий потери рабочего времени, связанные с профилактикой и ремонтом ЭВМ, примем $K_{пот}$ = 0,2.

Затраты на материалы ($З_{в.м}$), необходимые для обеспечения нормальной работы ПЭВМ вычисляются по формуле (Д.31):

$$З_{в.м} = \sum_{i} З_{пр i} \cdot \left(1 + K_{доп}\right) \cdot m_{i} \cdot K_{м.з}, \qquad (Д.31)$$

где $З_{пр}$ – затраты на приобретение (стоимость) ЭВМ, руб.;

$K_{доп}$ – коэффициент, характеризующий дополнительные затраты, связанные с доставкой, монтажом и наладкой оборудования, $K_{доп}$ = 12 – 13 % от $З_{пр}$;

$K_{м.з}$ – коэффициент, характеризующий затраты на вспомогательные материалы ($K_{м.з}$ = 0,01).

Затраты на текущий и профилактический ремонт ($З_{т.р}$) вычисляются по формуле (Д.32):

$$З_{т.р} = \sum_{i} З_{пр i} \cdot \left(1 + K_{доп}\right) \cdot m_{i} \cdot K_{т.р}, \qquad (Д.32)$$

где $K_{т.р}$ – коэффициент, характеризующий затраты на текущий и профилактический ремонт, $K_{т.р} = 0{,}05$.

Прочие затраты, связанные с эксплуатацией ЭВМ ($З_{пр}$) вычисляются по формуле (Д.33):

$$З_{пр} = \sum_i З_{прi} \cdot \left(1 + Д_{доп}\right) \cdot m_i \cdot K_{пр}\,, \qquad (Д.33)$$

где $K_{пр}$ – коэффициент, характеризующий размер прочих затрат, связанных с эксплуатацией ЭВМ ($K_{пр} = 0{,}05$).

Машинное время ЭВМ ($t_{эвм}$ в часах), необходимое для разработки и отладки проекта специалистом, вычисляется по формуле (Д.34):

$$t_{эвм} = \left(t_{р.п} + t_{вн}\right) \cdot F_{см} \cdot K_{см}\,, \qquad (Д.34)$$

где $t_{р.п}$ – срок реализации стадии «Рабочий проект» (РП);

$t_{вн}$ – срок реализации стадии «Ввод в действие» (ВП);

$t_{р.п} + t_{вн} = 60$;

$F_{см}$ – продолжительность рабочей смены, ч., $F_{см} = 8$ ч.;

$K_{см}$ – количество рабочих смен, $K_{см} = 1$.

Расчет затрат на изготовление эталонного экземпляра ($З_{эт}$) осуществляется по формуле (Д.35):

$$З_{эт} = \left(З_{тр} + З_{тех} + З_{мв}\right) \cdot K_{эт}\,, \qquad (Д.35)$$

где $K_{эт}$ – коэффициент, учитывающий размер затрат на изготовление эталонного экземпляра, $K_{эт} = 0{,}05$.

Затраты на материалы ($З_{мат}$) по формуле (Д.36):

$$З_{мат} = \sum_i Ц_i \cdot N_i \left(1 + K_{т.з}\right) - Ц_{0i} \cdot N_{0i}\,, \qquad (Д.36)$$

где $Ц_i$ – цена *i*-го наименования материала полуфабриката, комплектующего, руб.;

N_i – потребность в *i*-м материале, полуфабрикате, комплектующем, натур. ед.;

$К_{т.з}$ – коэффициент, учитывающий сложившийся процент транспортно-заготовительных расходов в зависимости от способа доставки товаров, $К_{т.з} = 0{,}1$;

$Ц_{0i}$ – цена возвратных отходов *i*-го наименования материала, руб.;

N_{0i} – количество возвратных отходов *i*-го наименования, натур. ед.;

n – количество наименований материалов, полуфабрикатов, и т.д.

Общепроизводственные затраты ($З_{общ.пр}$) рассчитываются по формуле (Д.37):

$$З_{общ.пр} = \frac{ЗП_{осн} \cdot Н_{доп}}{100}, \qquad (Д.37)$$

где $Н_{доп}$ – норматив общепроизводственных затрат.

Непроизводственные затраты рассчитываются по формуле (Д.38):

$$З_{непр} = \frac{ЗП_{осн} \cdot Н_{доп}}{100}, \qquad (Д.38)$$

где $Н_{непр}$– норматив непроизводственных затрат.

Оптовая цена программного продукта ($Ц_{опт}$) вычисляется с использованием формулы (Д.39):

$$Ц_{отп} = З_р + П_р, \qquad (Д.39)$$

Прибыль от реализации программного продукта ($П_р$) вычисляется с использованием формулы (Д.40):

$$П_р = \frac{З_р \cdot У_р}{100}, \qquad (Д.40)$$

где $З_р$ – себестоимость ПО, руб.;

$П_р$ – прибыль от реализации программного продукта, руб.;

$У_р$ – уровень рентабельности программного продукта, процентов ($У_р$ = 30 процентов).

Прогнозируемая отпускная цена программного продукта рассчитывается по формуле (Д.41):

$$Ц_{отп} = З_р + П_р + P_{ндс}, \quad (Д.41)$$

Налог на добавленную стоимость ($Р_{ндс}$) вычисляется по формуле (Д.42):

$$P_{ндс} = \frac{(З_р + П_р) \cdot Н_{ндс}}{100}, \quad (Д.42)$$

где $Н_{ндс}$ – ставка налога на добавленную стоимость, процентов, $Н_{ндс}$ = 20 процентов.

Годовой экономический эффект от производства нового программного обеспечения ($Э_{пр}$) определяется вычисляется с помощью формулы (Д.43):

$$Э_{пр} = (З_{пр.б} - З_{пр.н}) \cdot А_{пр.н} \quad (Д.43)$$

где $З_{пр.б}, З_{пр.н}$ – приведенные затраты на единицу выпуска ПО по базовому и новому вариантам, руб.;

$А_{пр.н}$ – годовой объем выпуска в расчетном году для реализуемого ПО, ед.

Срок окупаемости программного продукта, вычисленный с помощью формулы (Д.44):

$$О_{пп} = \sum_{i=1}^{n} \frac{Э_{пр}}{(1 + С_к)^i} - Ц_{опт}, \quad (Д.44)$$

где $С_к$ – ставка рефинансирования (равна 0,085);

$Ц_{опт}$ – отпускная цена программного продукта;

$Э_{пр}$ – годовой экономический эффект от производства нового ПП.

Таблица Д.1 – Технико-экономические показатели проекта

Наименование показателя	Единица измер.	Проектный вариант
коэффициент конкурентоспособности	–	1,86
Коэффициент эквивалентности	–	1,36
Коэффициент изменения функциональных возможностей	–	1,27
Коэффициент соответствия нормативам	–	1

Продолжение таблицы Д.1

Коэффициент цены потребления	–	0,93
Общая трудоемкость разработки ПО	чел.- дн	144
Затраты на оплату труда разработчиков	руб.	13169
Затраты машинного времени	руб.	252
Затраты на изготовление эталонного экземпляра	руб.	671
Затраты на технологию	руб.	–
Затраты на материалы	руб.	27,5
Общепроизводственные затраты	руб.	819
Непроизводственные (коммерческие) затраты	руб.	410
Суммарные затраты на разработку ПО ($З_р$)	руб.	15349
Оптовая цена ПП ($Ц_{опт}$) без НДС	руб.	19953
Оптовая цена ПП ($Ц_{опт}$) с НДС	руб.	23943
Экономический эффект от производства нового ПП	руб.	7875

На основании технико-экономических показателей проекта можно сделать вывод, что разработка программного продукта целесообразна.

ПРИЛОЖЕНИЕ Е
(рекомендуемое)
Список опубликованных работ

1. Расшивалов Н.И. Обнаружение плодов огурца для автоматизации процесса сбора многофункциональным аграрным роботом на основе сверточных нейронных сетей / Н.И. Расшивалов // Современные проблемы машиноведения : сб. науч. тр. В 2 ч. Ч. 1 / М-во образования Респ. Беларусь [и др.] ; под общ. ред. А. А. Бойко. – Гомель : ГГТУ им. П. О. Сухого, 2023. – С. 175–178.