Федеральное государственное автономное образовательное учреждение высшего образования «Казанский (Приволжский) Федеральный Университет»

На правах рукописи

Специальность: 2.2.11 – Информационно-измерительные и управляющие системы

Галиуллин Искандер Гаязович

ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ БЕСПИЛОТНЫХ СЕЛЬСКОХОЗЯЙСТВЕННЫХ ТРАНСПОРТНЫХ СРЕДСТВ

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель:
доктор технических наук,
директор ИИРСИ КФУ Д.Е. Чикрин

Казань – 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность исследования

Отечественная экономика сталкивается с вызовами современности, требует инновационных решений для повышения эффективности практически всех отраслей народного хозяйства. В последние десятилетия наблюдается нарастающее внедрение ресурсосберегающих технологий, информационно-измерительных и управляющих систем, средств автоматизации для решения широкого спектра производственных задач. Фактически революционным стало стремительное освоение автономных технологий и беспилотных транспортных систем отечественными промышленными предприятиями и научными организациями.

Создание и масштабное использование беспилотных сельскохозяйственных транспортных систем (БСТС) направлено не только на обеспечение технологического суверенитета и продовольственной независимости Российской Федерации, но и на сокращение трудозатрат и повышения производительности, а также на значительное снижение негативного воздействия на окружающую среду.

Создание современных БСТС требует не только разработки самих систем, но и учета ряда методологических аспектов. Важность комплексного проектирования, включая применение концепции бережливого производства и шаблонов проектирования, высока, поскольку эффективное внедрение БСТС зависит от создания технологических, экологически устойчивых и оптимальных решений.

Таким образом, диссертация посвящена исследованию и разработке современных сельскохозяйственных БСТС, с фокусом на сценариях эксплуатации, уровнях использования современных методологий проектирования информационно-управляющих систем (ИС) для создания эффективных и устойчивых технологических решений в сельском хозяйстве.

При подготовке исследования были рассмотрены научные работы связанные с перспективами развития беспилотной сельскохозяйственной

техники, интеллектуальными системами в сельском хозяйстве, повышением уровня интеграции беспилотных технологий в эксплуатацию сельскохозяйственной техники и перспективами применения беспилотного транспорта в России за авторствами: Каратаевой О.Г., Виноградова О.В., Харламова Д.И., Митенева Н.С., Алексеев Ю.М. [1], Пономарева Д.А., Шияновой К.С.[2], Кацуба Ю.Н., Караваева Н.А. [3], Бондаревой А.А. Паршиной Л.Н. [4] и др.

Научная проблема

Научная проблема, освещаемая в данной диссертационной работе, сосредотачивается вокруг комплексного исследования и разрешения ключевых аспектов внедрения беспилотных сельскохозяйственных транспортных систем (БСТС). Основной научной проблемой является отсутствие комплексного понимания и эффективных решений для создания, оптимизации и интеграции инфокоммуникационных систем в сельскохозяйственную технику, что ограничивает их полноценное внедрение и воздействие на современное сельское хозяйство.

Исходя из обозначенных проблем и актуальности исследования, выделим объект и предмет исследования, сформулируем цели и задачи.

Объект исследования

Объектом исследования являются современные беспилотные сельскохозяйственные транспортные системы и инфокоммуникационные компоненты, входящие в их структуру.

Предмет исследования

Предметом исследования являются процессы проектирования, оптимизации и интеграции этих систем в сельскохозяйственную деятельность.

Цель работы

Повышение уровня интеграции информационно-управляющих процедур в процессы создания и эксплуатации БСТС на основе разработки шаблонов их проектирования с определением и устранением ключевых системных ограничений.

Задачи исследований

Для достижения поставленной цели определяются следующие задачи:

1. Исследование возможностей и разработка методов адаптации существующих шаблонов проектирования в автомобильной промышленности к транспортным средствам сельскохозяйственного назначения на основе анализа существующих и перспективных беспилотных сельскохозяйственных транспортных систем.

2. Анализ и выявление системных ограничений информационно-измерительных и управляющих систем, интегрированных в сельскохозяйственные транспортные средства, для их последующего устранения и повышения эффективности работы системы.

3. Разработка методов устранения системных ограничений на основе концепции бережливого производства и теории ограничений Элияху Голдратта для повышения эффективности функционирования беспилотных сельскохозяйственных транспортных систем.

4. Разработка методов виртуального моделирования и проведение полунатурных испытаний разработанных информационно-управляющих систем для обеспечения оптимизации проектирования и оценки их показателей.

Степень успешного решения данных задач определит теоретическую ценность работы, предоставляя новые знания в области беспилотных систем для сельского хозяйства, а также ее практическую ценность, создавая основу для эффективного внедрения инноваций в данную отрасль.

Методы исследований

В процессе научного исследования были получены результаты через анализ и систематизацию последних достижений в области сельскохозяйственной автоматизации; сравнительный анализ для выявления различий и сходств между существующими беспилотными системами, а также для определения их преимуществ и недостатков. Теоретическая база работы формировалась с использованием методов математического и виртуального

моделирования, обработки изображений, а также технологий распознавания образов и фильтрации цифровых данных.

Кроме того, применялись принципы бережливого производства и теории ограничений для рассмотрения вопросов оптимизации процессов проектирования инфокоммуникационных систем беспилотных транспортных средств с целью повышения эффективности.

Экспериментальная составляющая включала в себя натурные испытания и применение программного обеспечения для математического и системного моделирования.

Научная новизна диссертации

1. Разработан универсальный шаблон проектирования беспилотных сельскохозяйственных транспортных систем, состоящий из 4 подсистем и 14 функциональных блоков, что позволяет создать новые элементы структуры информационно-измерительных и управляющих систем для сельскохозяйственного назначения.

2. В работе впервые выявлены и описаны ключевые системные ограничения ин-формационно-управляющих систем беспилотных сельскохозяйственных транспортных средств. Это значимо для понимания особенностей функционирования данных систем и определения направлений их улучшения. Ключевые системные ограничения:

- ограниченность ресурсов и строгая приоритезация задач вычислительной подсистемы (латентность срабатывания на уровне 0,5-1 сек; количество одновременно выполняемых задач не более 2-х).

- пропускная способность, латентность, джиттер, помехоустойчивость подсистемы коммуникации и связи (скорость связи без потери соединения не более 1-2 Мбит\с, латентность на уровне 300-400 мс, джиттер на уровне 20-50 мс).

- частота получения навигационных оценок, точность позиционирования, качество сигнала и количество наблюдаемых спутников в подсистеме позиционирования и навигации (частота получения

навигационных оценок 1Гц, точность позиционирования – 2-3 метра, обеспечивается достаточное качество приема сигналов спутников исключительно на открытой местности, без заезда в укрытия гаражного\ангарного типа).

- скорость обработки данных, количество и качество различных распознаваемых объектов в подсистеме машинного зрения (количество обрабатываемых кадров – до 2 в секунду, количества одновременно анализируемых объектов – от 3 до 5; стабильная работа исключительно в условиях отсутствия засветки и неблагоприятных погодных явлений).

3. Предложены новые методы устранения системных ограничений в сельскохозяйственных беспилотных транспортных системах, основанные на синергии бережливого производства и теории ограничений. Этот подход является инновационным и позволяет эффективно оптимизировать процессы функционирования данных систем:

- вычислительная подсистема: использование операционной системы реального времени совместно с индустриальными микроконтроллерами с приоритезацией задач с использованием расширения реального времени обеспечивает латентность срабатывания на уровне 0,05-0,1 сек с количеством одновременно выполняемых задач – до 14 с динамической регулировкой приоритетов (соответственно количеству функциональных блоков информационно-управляющих системы).

- система позиционирования и навигации: использование многоточечной системы инерциальной навигации одновременно с методом получения высокоточных координат и взаимной калибровкой показаний обеспечивает повышение частоты обновления навигационных оценок до 100 Гц с уменьшением СКО позиционирования до 0,1 метра. Одновременно с этим достигается функционирование системы навигации в полном автономе (в условиях недоступности навигационного созвездия спутников ГНСС) до 10-15 минут с точностью до 1,5% от пройденной траектории в указанном режиме.

- система связи: использование низколатентных профилей радиоинтерфейсов 5G одновременно с применением протокола помехоустойчивой связи без установления соединия позволяет радикально (в 10 и более раз) снизить количество разрывов (потерь пакетов, сопровождающих повреждения и-или комплексную потерю видеокадра) при скоростях передачи до 10-20 Мбит\с.

- машинное зрение: использование современных аппаратно-акселерированных нейросетей последнего поколения, оптимизированных по размеру видеопамяти и раз-решению совместно с унификацией условий наблюдения при помощи алгоритмов адаптивной нормализации видеопотока обеспечили увеличение показателя количества кадров обработки до 10 в секунду и количества одновременно анализируемых обьектов от 15 до 20, в том числе при неблагоприятных условиях наблюдения (снег, дождь, туман, сумерки и пр.)

4. Разработана новая структура и методы построения виртуальных полигонов – верифицируемых систем достоверной симуляции и комплексного воспроизведения динамических процессов и объектов робототехнических транспортных систем сельскохозяйственного назначения. Это новшество позволяет создавать более точные и реалистичные модели данных систем для проверки и верификации их работы: наивысшее процентное соотношение максимального отклонения по отношению к общей длине траектории составляет 0,42%, а СКО по отношении к длине траектории: 0,14% (сравнение маршрутов движения реального и виртуального БСТС).

Научные положения, выносимые на защиту

На защиту выносятся следующие научные положения, выдвигаемые на основе полученных в диссертационной работе результатов:

1) Разработаны шаблоны проектирования информационно-управляющих систем беспилотных сельскохозяйственных транспортных систем (соотв. п.п. 2,3 паспорта специальности 2.2.11).

2) Определены системные ограничения БСТС и их отдельных подсистем, включая вычислительную подсистему, подсистему машинного зрения, подсистему навигации и ориентации и др. (соотв. п.п. 2,6 паспорта специальности 2.2.11).

3) С использованием концепции бережливого производства и теории ограничений Элияху Голдратта определены методы снятия ранее найденных системных ограничений (соотв. п.п. 1,4 паспорта специальности 2.2.11).

4) Разработан инструментарий системы физико-технического виртуального моделирования БСТС и их отдельных подсистем для целей оптимизации скорости построения и оценки показателей разрабатываемых информационно-управляющих систем (соотв. п.п. 7,8 паспорта специальности 2.2.11).

Практическая ценность диссертации

1). Разработанные шаблоны проектирования обеспечивают ускорение и значительную степень унификации технических решений при разработке систем беспилотных транспортных средств.

2). Принципы определения системных ограничений позволяют обнаружить ограничения на стадии проектирования и обеспечить учет их при принятии конкретных технических решений.

3). Идентификация и снятие системных ограничений способствует более быстрому и точному принятию решений в ходе разработки инфокоммуникационных систем. Это ускоряет внедрение новых технологий и снижает временные затраты.

4). Подходы виртуального и полунатурного моделирования предлагают инструменты для более эффективной верификации разработанных систем. Это уменьшает риски при внедрении новых технологий и способствует созданию более надежных и адаптируемых беспилотных сельскохозяйственных транспортных средств.

Личный вклад автора

Автор внес значительный личный вклад в исследование, представленное в данной диссертации. Основные достижения и вклад заключается в:

- разработке и успешной реализации инновационных шаблонов проектирования, что существенно обогатило методологию разработки информационно-управляющих систем беспилотных сельскохозяйственных транспортных средств;

- идентификации и детальном анализе ключевых системных ограничений в беспилотных сельскохозяйственных транспортных системах, что создало основу для последующего снятия данных ограничений;

- разработке эффективных методов устранения системных ограничений, основанных на синергии концепции бережливого производства и теории ограничений, что предоставило новые принципы и их успешную практическую проверку;

- создании инновационной структуры и методов построения виртуальных полигонов для верификации систем достоверной симуляции беспилотных сельскохозяйственных транспортных систем;

- руководстве группой специалистов и инженеров КФУ, занимающихся низкоуровневой разработкой и автоматизацией основных узлов тракторов малого и тяжелого классов;

- личное участие и руководство при построении архитектуры, разработке алгоритмов, написании программ и отладке программного обеспечения на целевой сельскохозяйственной технике;

- руководстве проведением испытаний систем с целью получения данных для проведения верификации работоспособности и эффективности разработанных систем.

Результаты внедрения

Результаты работы внедрены в деятельность ООО «ТПК МТЗ-Татарстан» при проведении работ по автоматизации тракторов Беларус-112Н, Беларус-3522.

Материалы диссертационной работы внедрены и использованы в Научно-исследовательском центре «Центр превосходства Специальная робототехника и искусственный интеллект» Института вычислительной математики и информационных технологий для разработки серии беспилотных сельскохозяйственных наземных транспортных средств при получении ключевых результатов Центра по программе Приоритет-2030.

Указанные внедрения подтверждены актами.

Публикации

Основные научные и практические результаты работы представлены в 7 статьях в изданиях из списка ВАК по специальности, результаты интеллектуальной деятельности защищены 13 свидетельствами о государственной регистрации программы для ЭВМ.

Апробация результатов работы

Основные положения и результаты диссертационного исследования докладывались, обсуждались и представлялись на следующих конференциях:

- Научно-техническая конференции с международным участием имени профессора О.Н. Пьявченко “Компьютерные и информационные технологии в науке, инженерии и управлении” (КомТех-2023), г. Таганрог, 2023 г.

- XVI Всероссийская мультиконференция по проблемам управления (МКПУ-2023), Робототехника и мехатроника (РиМ-2023), г. Волгоград, 2023 г.

- Одиннадцатая всероссийская научно-практическая конференция по имитационному моделированию и его применению в науке и промышленности «Имитационное моделирование. Теория и практика» (ИММОД-2023), г. Казань, 2023 г.

- Мультидисциплинарный международный форум «Kazan Digital Week – 2022», «Kazan Digital Week – 2023», г. Казань, 2022, 2023 гг.

Глава 1. Сценарии эксплуатации, уровни автоматизации техники в сельском хозяйстве и известные реализации.

1. Существующие и перспективные БСТС в сельском хозяйстве

1.1. Классификация БСТС

Основным мировым трендом в сельском хозяйстве последнего десятилетия является точное (координатное) земледелие (precision farming). Точное земледелие предусматривает управление агротехнологическими операциями с максимальным учетом внутриполевой неоднородности почвенного покрова и состояния посевов сельскохозяйственных культур. Технические средства дифференцированно производят технологические операции с почвой и посевами на основе точного анализа информации в пределах обрабатываемого поля [5].

К техническим средствам аналитики сельскохозяйственного назначения относят:

а) беспилотные транспортные наземные средства;

б) беспилотные летательные аппараты (БПЛА);

в) полевые датчики и сенсоры, объединенные в сложные, взаимодействующие сети.

Основным средством решения задач точного земледелия по-прежнему остаются наземные средства. Именно они будут являться носителями интеллектуального перспективного навесного оборудования, которое будет совмещено с программным обеспечением.

В Таблице 1.1 представлены уровни автономности, по которым классифицируется наземная сельскохозяйственная техника по степени выполнения технологических операций [6].

Таблица 1.1 - уровни автономности сельскохозяйственной техники.

Уровень автономности	0	1	2	3	4	5
	Традиционное земледелие	Технологии точного земледелия	Координация и оптимизация	Автономность в режиме реального времени	Контролируемая автономность	Полная автономность
Описание	 Автоматизация не используется	 Добавлена какая-либо автоматизированная система к оборудованию	 Координация и синхронизация полевых задач на нескольких машинах	 Автоматизация техники в режиме реального времени в зависимости от условий окружающей среды	 Техника выполняет операции без прямого участия человека.	 Полностью автономное сельское хозяйство будущего
Автоматизация	Ручное управление техникой	Уровень задачи - каждая технология работает независимо	Уровень техники - для каждой техники требуется оператор, однако технологии взаимодействуют между собой и выполняют раздельные задачи	Уровень окружения - оператор присутствует на технике, но технологии полностью управляют машиной	Полевой уровень – операции техникой выполняются без участия человека, однако при его присутствии.	Уровень предприятия – управление удаленно.

Исследования автора посвящены построению систем автоматизации транспортных средств сельскохозяйственного назначения.

1.1.1. Средства автоматизации и типы наземной сельскохозяйственной техники с функционалом БСТС

На текущий момент времени, автоматизация сельскохозяйственной техники сводится к неразрушающему внедрению в её структуру следующих технических средств [7]:

Модуль курсоуказания – техническое средство, представляющее собой бортовой компьютер, совмещающий в себе функции дисплея и блока принятия решений. С модулем курсоуказания сопрягаются, в частности, навигационный

модуль, а также иное бортовое оборудование, которым оснащается единица сельхозтехники [8].

Навигационный модуль – техническое средство, формирующее актуальные измерительные данные о геодезических координатах, скорости и пр. На основании сформированных навигационным модулем измерительных данных, программное обеспечение, заложенное в модуль курсоуказания, осуществляет принятие решения о генерации команд управления бортовым оборудованием, установленным на борту единицы сельхозтехники [9].

Модули систем параллельного вождения – технические средства, внедряемые в систему рулевого управления и осуществляющие помощь оператору, путём коррекции угла поворота рулевого колеса. Решение о внесении коррекции в управление принимается с помощью модуля курсоуказания, на основании измерительных данных, полученных от навигационного модуля [10].

Модули систем автопилотирования – технические средства, внедряемые в систему рулевого управления, трансмиссии и систему управления силовой установкой единицы сельхозтехники, и позволяющие ей функционировать, по заложенной в модуль курсоуказания программе автономно, без участия человека [11].

Усложнение сельскохозяйственных машин, условий их использования, повышающиеся требования к качеству выполнения технологического процесса вызывают необходимость использования новых подходов и концепций. Они базируются на применении современных информационных технологий, автоматизированных систем контроля и управления технологическими процессами, глобальных систем позиционирования.

В конструкциях современных тракторов реализуется технические решения, способствующие повышению технико-экономических и экологических показателей, улучшению управления машинно-тракторными агрегатами и созданию удобств механизаторов. При этом одна из основных тенденций – внедрение информационных и управляющих систем на основе

электроники, обеспечивающих минимальное вмешательство оператора в управление машинно-тракторным агрегатом [12].

Внедрение информационных систем на современных зерноуборочных комбайнах осуществляется в основном в следующих направлениях: контроль и регулирование параметров двигателя, частоты вращения рабочих органов; автоматическое регулирование технологической загрузки; контроль потерь зерна за молотильно-сепарирующим устройством и уровня заполнения бункеров; контроль и регулирование высоты среза, давления жатки на почву; копирование рельефа поля; программирование технологических настроек комбайна на уборку определенной культуры; автоматическое вождение комбайна; поиск и диагностирование неисправностей.

Современные системы управления и контроля технологического процесса находят применение на таких сложных машинах, как свеклоуборочные и картофелеуборочные комбайны.

1.1.2. Сравнение возможностей средств автоматизации различных производителей и обзор современных российских сельскохозяйственных БСТС

Основными фирмами, представляющими оборудование на рынке точного фермерства, являются Ag Leader (США), AGCO Corporation (США), CropX (США), John Deere (США), Trimble, Inc. (США), Leica Geosystems (Швейцария), CLAAS (Германия) [13].

В России в настоящее время функционируют свои системы параллельного вождения:

• навигационный пульт «Азимут-1» от компании ООО «Ратеос»;

• системы COMMANDER и Атлас 730 от компании ООО «КСМ-Интех»;

• система АГРОНАВИГАТОР от компании ООО «ЦТЗ Аэросоюз».

• система РСМ Агротроник Пилот 1.0 и Пилот 2.0 от ООО «Ростсельмаш»

Компания Cognitive Technologies совместно с производителем сельхозтехники «Ростсельмаш» и агрохолдингом «Союз-Агро» осуществляет разработку беспилотной сельхозтехники. В 2016 году прошли первые испытания трактора с системой компьютерного зрения C-Pilot [14].

В Кабардино-Балкарском научном центре РАН разработан мультиагентный робот комбайн для уборки урожая плодоовощной продукции на открытом грунте в «безлюдном» режиме – «AgroMultiBot.Garnet», который стал первым в составе семейства роботов для безлюдного сельскохозяйственного производства, разрабатываемого группой компаний N’art Robotix. [15].

В России на данный момент компанией «Ростсельмаш» испытывается беспилотный комбайн - TORUM 785. Беспилотный комбайн – часть инновационного проекта. Проект направлен на массовое использование беспилотных технологий и высокоточной навигации в сельском хозяйстве [16].

В Таблице 1.2 представлен сравнительный анализ возможностей и характеристик различных российских БСТС и средств автоматизации.

Таблица 1.2 — сравнительный анализ возможностей и характеристик различных российских БСТС и средств автоматизации.

Основные характеристики	**Агронавигатор Agroglobal стандарт**	**РСМ Агротроник Пилот 1.0 и Пилот 2.0**	**Cognitive Agro Pilot**	**AgroMultiBot**	**TORUM 785**
Точность навигации	2-20 см	2,5 см	2-5 см	до 20 см	2,5 см
Возможность выполнения различных задач (плугование, посев, опрыскивание и пр.)	да	да	да	да	только уборка зерновых культур
Обнаружение и определение искусственных и естественных препятствий	нет	да	да	нет	да

Необходимость RTK для работы в поле	да	нет	нет	да	да
Движение по кромке/валкам	нет	да	да	нет	нет
Независимость от спутниковой ситуации	только машина с подготовкой	да	да	нет	нет
Необходимость наличия цифровой шины CAN	нет	да	да	-	-
Управление скоростью	нет	да	да	да	да
Автоматическое определение поворотов/разворотов	нет	да	да	нет	да
Отправка телеметрии	нет	нет	да	нет	да
Наличие машинного зрения	нет	да	да	нет	да
Функционирование в ночное время суток	нет	да	да	нет	не подтверждено
Работа в различных погодных условиях	нет	да	да	нет	не подтверждено

1.1.3. Сценарии эксплуатации современных и перспективных БСТС

В соответствии с существующими на рынке и перспективными испытываемыми решениями, возможно выделить 4 обобщенных сценария эксплуатации [17] БСТС, имеющих отношение к реализации технологий высоких уровней автоматизации БСТС:

– **Обработка почвы – вспахивание и культивирование**:

Внедрение точного земледелия начинается с экономного движения техники и правильного использования сельскохозяйственных агрегатов. Это касается культивирования и вспахивания без пропусков и повторных проходов. При этом БСТС должна выполнять все почвообрабатывающие операции с возможностью объезда различных препятствий, таких как столбов, кустарников, камней, островков и т.д.

– **Уборка урожая**:

Сценарий эксплуатации беспилотных комбайнов для уборки урожая включает в себя автоматическое движение комбайна по полю, выгрузку собранного урожая в контейнер и возвращение обратно на поле. Беспилотный комбайн может быть оснащен различными датчиками, которые позволяют определять качество собранного урожая и его количество.

– **Посев**:

Сценарий эксплуатации БСТС совместно с агрегатами для посева включает в себя автоматическое движение по полю, определение оптимальной глубины посева и скорости движения. Беспилотная система может быть оснащена датчиками, которые позволяют определять качество и количество посева.

– **Опрыскивание**:

Сценарий эксплуатации беспилотных БСТС совместно с опрыскивающими агрегатами для обработки полей включает в себя автоматическое движение по полю, определение оптимального расхода жидкости и скорости движения. Беспилотная система может быть оснащена датчиками, которые позволяют определять качество и количество жидкости, а также определять необходимость повторной обработки.

Преимущества эксплуатации беспилотных сельскохозяйственных транспортных средств включают:

1. Увеличение производительности. Беспилотные транспортные средства позволяют сократить затраты на трудовые ресурсы и увеличить производительность.

2. Повышение точности и качества работ. Беспилотные транспортные средства оснащены датчиками, которые позволяют определять оптимальные параметры работ и качество выполненных работ.

3. Снижение затрат на топливо и обслуживание. Беспилотные транспортные средства потребляют меньше топлива, чем обычные транспортные средства, и требуют меньше обслуживания.

4. Сокращение рисков для здоровья работников. Беспилотные транспортные средства позволяют избежать рисков для здоровья работников, связанных с работой в условиях повышенной опасности.

5. Снижение негативного влияния на окружающую среду. Беспилотные транспортные средства потребляют меньше топлива и не выбрасывают вредные вещества в атмосферу, что позволяет снизить негативное влияние на окружающую среду.

В целом, беспилотные сельскохозяйственные транспортные средства имеют большой потенциал для решения многих задач в сельском хозяйстве.

1.1.4. Требования и основные направления создания современных сельскохозяйственных БСТС

Основное требование к техническим средствам обеспечения решения задач точного земледелия – высокая планово-высотная точность полей с ежегодной повторяемостью результатов (в плане – до 1 см, по высоте до 5 см). Еще одним условием является готовность к работе в сложных условиях – часто требуется решение задач в различных климатических условиях, во время продолжающихся дождей, тумана, сумерек и др. Для протяженных полей и частых остановок по пути следования требуется сравнительно высокая скорость движения между заданными точками маршрута [18]. В пределах поля и между обрабатываемыми полями необходимо поддерживать устойчивую связь между БСТС и оператором.

Обобщая требования со стороны сельскохозяйственной практики, сформулируем основные технические требования к БСТС:

– высокая точность позиционирования (не хуже 10-20 см);

– сравнительно высокая скорость движения (не менее 15-20 км/час);

– запас топлива не менее, чем на 5-6 часов функционирования;

– работоспособность в сложных условиях (дождь, грязь, распутица, сумерки и др.), в идеале в формате 24/7/365;

– иметь систему технического зрения для движения в сложных геофизических условиях, в том числе по пересеченной местности;

– иметь беспроводную связь с оператором; при отсутствии связи функционировать автономно до восстановления связи;
– формировать и корректировать с помощью оператора (или самостоятельно при автономном функционировании) маршрут движения;
– реализовывать решение задач распознавания ситуаций, возникающих в среде обитания растений, с отнесением ситуаций к определенному классу;
– иметь возможность установки разнообразного навесного оборудования для решения задач точного земледелия (дифференцированное внесение удобрений, борьба с сорняками, вредителями и др.);
– быть легко ремонтопригодным, обслуживаться минимальным количеством специалистов.

Анализ перечисленных технических требований по существу определяет основные направления создания сельскохозяйственных БСТС.

2. Системы автоматизации БСТС

2.1. Состав и особенности систем автоматизации БСТС

В современном мире беспилотные сельскохозяйственные транспортные средства играют важную роль в автоматизации процессов сельского хозяйства. Они позволяют увеличить производительность, улучшить качество работы и снизить затраты на труд и ресурсы. Для управления БСТС используются различные системы автоматизации, которые включают в себя различные компоненты:

- Датчики

Датчики являются одним из основных компонентов систем автоматизации БСТС. Они предназначены для сбора информации о состоянии окружающей среды и параметрах транспортного средства, таких как скорость, ускорение, угол наклона, температура и т.д. Датчики могут быть различных типов, включая геодезические, инерциальные, оптические и другие.

Также они могут включать в себя камеры видеонаблюдения, радары, лидары и другие сенсоры, которые помогают предотвращать столкновения с препятствиями, а также обеспечивать безопасное использование БСТС.

- Контроллеры

Контроллеры (управляющие блоки) – это электронные устройства, которые принимают информацию от датчиков, обрабатывают ее и выдают команды актуаторам. Контроллеры выполняют основные функции анализа данных, принятия решений и управления работой всех компонентов системы автоматизации.

Контроллеры беспилотных транспортных средств с/х назначения могут иметь различную аппаратную и программную архитектуру. Например, это может быть централизованный контроллер, который управляет всеми подсистемами БСТС, либо распределенные контроллеры, которые работают независимо друг от друга и обмениваются информацией через шину данных.

- Актуаторы

Актуаторы – это компоненты системы автоматизации, которые управляют физическими процессами в беспилотном транспорте. Они позволяют контролировать движение транспорта, изменять направление движения, скорость и выполнять другие действия.

Для управления движением транспорта могут использоваться различные типы актуаторов, например, электромоторы, гидравлические механизмы, пневматические механизмы и др. Выбор конкретного типа актуатора зависит от конкретной задачи и требований к управлению.

- Системы связи

Системы связи позволяют беспилотным системам передавать данные между собой и с операторами, что позволяет операторам удаленно управлять транспортными средствами и контролировать их работу. Для связи могут использоваться различные технологии, включая радиочастотную связь, сотовую связь, спутниковую связь или другие. Выбор технологии связи

зависит от многих факторов, таких как дальность связи, пропускная способность и стоимость.

- Система позиционирования и навигации

Еще один важный компонент – это система позиционирования и навигации. Она используется для определения местоположения БСТС на поле, а также для планирования маршрута и автоматической корректировки движения транспортного средства. Обычно в состав системы позиционирования и навигации входят GPS-приемник, инерциальные датчики и другие устройства.

- Система машинного зрения

Система машинного зрения является одной из ключевых компонент системы автоматизации БСТС. Она представляет собой совокупность аппаратных и программных средств, способных воспринимать и анализировать визуальную информацию из окружающей среды. Основной целью системы машинного зрения является обнаружение, распознавание и анализ объектов и препятствий на пути движения транспортного средства. Для достижения этой цели система машинного зрения использует различные техники и алгоритмы компьютерного зрения. В состав системы машинного зрения могут входить различные компоненты, такие как камеры, радары, сонары и другие аппаратные средства. Программное обеспечение системы машинного зрения выполняет обработку полученных изображений и применяет различные алгоритмы компьютерного зрения для обнаружения и распознавания объектов. Эти алгоритмы могут включать в себя фильтрацию изображений, сегментацию объектов, извлечение признаков и классификацию.

Одной из особенностей системы машинного зрения является способность работать в реальном времени, обеспечивая оперативную обработку и анализ визуальной информации. Это позволяет БСТС быстро реагировать на изменения в окружающей среде и принимать соответствующие решения.

2.2. Структура инфокоммуникационных систем транспортного средства

Инфокоммуникационные системы БСТС представляют собой сложную структуру, которая включает в себя различные устройства, сенсоры, датчики, системы связи и программное обеспечение. Эти компоненты обеспечивают сбор, передачу и анализ информации о состоянии и процессах внутри транспортного средства.

В настоящее время БСТС представляют собой комплексную систему автоматизации, состоящую из нескольких инфокоммуникационных подсистем:

1. Позиционирование и навигация: обеспечивает определение местоположения БСТС и точное определение времени.

2. Машинное зрение: использует различные датчики и камеры для создания трехмерной модели окружающего пространства и обнаружения препятствий.

3. Коммуникация и связь: обеспечивает передачу данных между компонентами системы и обеспечивает связь с внешней средой.

4. Вычислительная подсистема БСТС: включает в себя систему телеуправления, поддержку принятия решения и автопилотирования. Она отвечает за обработку данных и принятие решений на основе анализа информации, полученной от других подсистем БСТС. Более того, вычислительная подсистема обеспечивает связь между различными подсистемами и координирует их работу.

Общая структурная схема ИС БСТС представлена на Рисунке 1.1.

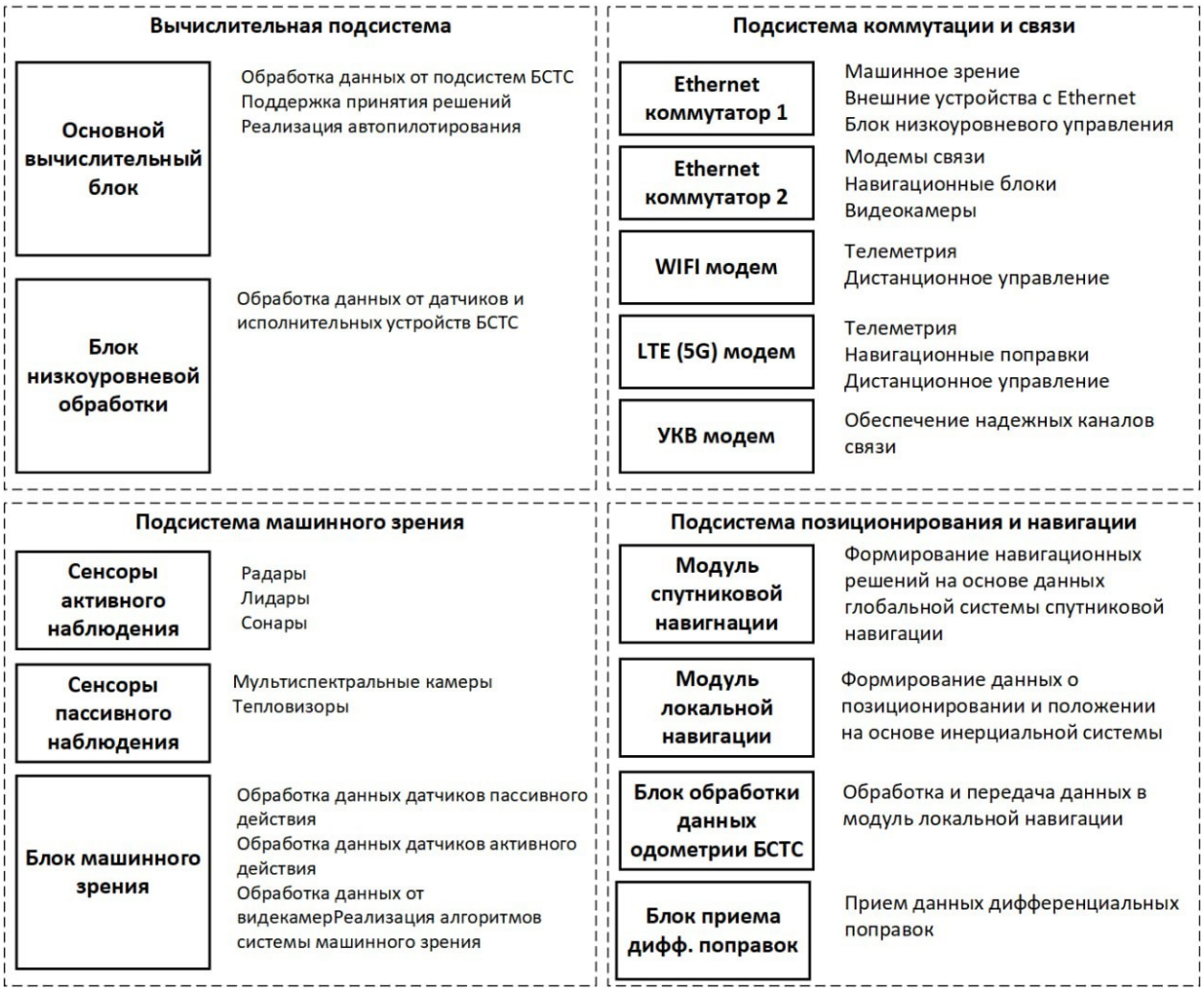


Рисунок 1.1 — Общая структурная схема ИС БСТС

В целом, ИС БСТС является сложной системой, требующей комплексного подхода к проектированию и разработке. Каждая из подсистем имеет свои особенности и требует специальных знаний и навыков для проектирования и интеграции в общую систему.

3. Методологии проектирования ИС БСТС

Выбор методологии проектирования является критически важным этапом в процессе разработки ИС, так как от него зависит качество итогового продукта, его возможность к дальнейшему развитию и масштабированию [19].

Однако, в рамках проектирования БСТС существует множество различных подходов и методов, что часто приводит к разобщенности существующих решений. Чтобы бороться с этой проблемой, предлагается

комплексная методология проектирования, которая позволяет использовать наработки и подходы из смежных отраслей промышленности. Также рассматриваются особенности стандартных методологий проектирования [20, 21], и показывается, что использование отдельно взятых методологий неприемлемо при проектировании БСТС, поскольку они не учитывают все специфические особенности данной области.

В результате, на основе анализа стандартных методологий и требований к проектированию БСТС, была разработана совмещенная методология проектирования, которая позволяет учесть все необходимые аспекты и обеспечить высокое качество итогового продукта.

3.1. Современные практики использования методологий проектирования в ИС БСТС

Исследование показало, что существует связь между методологией проектирования и качеством конечной системы, особенно при проектировании программного обеспечения [22]. Несмотря на то, что современные БСТС являются сложной совокупностью программных и аппаратных средств, они так же построены по тем же принципам, что и другие сложные программные комплексы [22, 23].

Существует несколько методологий проектирования, которые могут быть применены для создания беспилотных сельскохозяйственных транспортных средств:

1. Методология «сверху-вниз» [24].
2. Методология «снизу-вверх»[25].
3. Методология проектирования «V-модель» (или Vee) [26].
4. Методология H-GQM[27].

В источниках [28, 29] эти методологии рассмотрены в контексте синтеза и верификации ИС беспилотных транспортных средств на базе формализованных шаблонов проектирования. Далее автором предлагается

применение данных шаблонов для проектирования ИС БСТС в соответствии с концепцией бережливого производства.

3.2. Бережливое производство в контексте проектирования ИС БСТС

Ранее разработка ИС БСТС часто осуществлялась без использования шаблонов и методологий проектирования. Команды разработчиков полагались на индивидуальный опыт и творческий подход к созданию системы. Однако такой неструктурированный подход имел свои недостатки и ограничения, и мог привести к недооценке сложности системы, неправильному планированию и неэффективному использованию времени и ресурсов. Отсутствие шаблонов и методологий ведет к неоднородности и несогласованности в процессе разработки.

Оптимальным подходом к разработке ИС БСТС является применение концепции бережливого производства, которая предлагает систематический и структурированный подход к разработке, основанный на принципах минимизации потерь, повышения качества и непрерывного улучшения системы.

Концепция бережливого производства представляет собой философию и методологию, которая применяется в области разработки и проектирования продуктов с целью повышения эффективности, качества и сокращения времени разработки. Она основана на принципах и инструментах бережливого производства, разработанных в японской производственной системе Toyota.

Основная идея бережливого производства заключается в устранении неэффективности, избыточности и потерь в процессе разработки продукта. Она ставит акцент на создании ценности для клиента, сокращении времени цикла разработки, устранении избыточной работы, повышении гибкости и обеспечении качества.

Основные принципы бережливого производства включают:

1. Фокус на клиента: ориентация на потребности и требования клиента является основой для разработки продукта. Понимание ценности,

которую продукт предоставляет клиенту, помогает определить ключевые характеристики и функциональность.

2. Устранение потерь: идентификация и устранение всех видов потерь, таких как избыточная работа, ожидание, переработка, перенос задач и другие, которые могут возникнуть в процессе разработки продукта.

3. Постепенность: разработка продукта проводится поэтапно, с постепенным добавлением функциональности и проверкой результата на каждом этапе. Это помогает своевременно выявлять и исправлять ошибки и изменения.

4. Постоянное улучшение: концепция бережливого производства подразумевает постоянное стремление к улучшению процессов разработки и проектирования. Применение инструментов, таких как Kaizen (непрерывное улучшение), позволяет выявлять проблемы и находить эффективные решения для повышения производительности.

5. Кросс-функциональность и командная работа: принципы бережливого производства способствуют сотрудничеству и командной работе между различными функциональными областями, такими как инженеры, дизайнеры, производственные специалисты и другие. Это позволяет более эффективно использовать ресурсы и экспертизу каждого участника процесса.

Бережливоге производство в контексте разработки ИС БСТС может быть применен для улучшения эффективности и качества процесса разработки, а также оптимизации производительности и устойчивости системы. В Таблице 1.3 указаны некоторые аспекты бережливого производства, которые можно рассмотреть в данном контексте:

Таблица 1.3 — аспекты бережливого производства применительно к разработке ИС БСТС.

Наименование подхода	**Описание**	**Применимость с точки зрения разработки ИС БСТС**
Устранение избыточности	Нацелен на устранение неэффективных и избыточных операций	Удаление ненужных функций, упрощение алгоритмов или

		сокращение лишних шагов в процессе разработки
Управление потоком работ	Подразумевает плавный и эффективный поток работ между различными участниками и этапами разработки	Может включать оптимизацию коммуникации и сотрудничества между разработчиками программного обеспечения, инженерами аппаратного обеспечения и специалистами в области сельскохозяйственной техники.
Участие клиента	Подразумевает активное участие клиента в процессе разработки	Тесное взаимодействие сельскохозяйственных предприятий или фермеров, которые будут использовать систему автоматизации. С их помощью можно определить реальные потребности и требования, чтобы система соответствовала их ожиданиям и работала наилучшим образом в сельскохозяйственной среде.
Контроль качества	Включает в себя постоянный контроль качества и постепенное улучшение процессов	Может включать тестирование и проверку функциональности системы на различных этапах разработки, чтобы обнаружить и исправить возможные ошибки или недочеты.
Инкрементальное развертывание	Пошаговое внедрение изменений и обновлений в процессе разработки	Постепенное добавление функций автопилота, начиная с базовых возможностей и постепенно расширяя их в зависимости от потребностей и обратной связи от пользователей.
Управление рисками	Управления рисками и минимизации потенциальных проблем	Проведение анализа рисков и принятие мер для их снижения. Такие меры включают резервирование времени и ресурсов на разработку и испытания, а также создание системы обратной связи для обнаружения проблем и

		быстрого реагирования на них.
Стратегия непрерывного улучшения	Постоянное совершенствование процессов и продукта	Проведение постоянных анализов и обновлений, основанных на обратной связи от пользователей и новых технологических достижениях.
Эффективное использование ресурсов	Оптимизация использования ресурсов, включая время, трудовые затраты и материальные ресурсы	Рациональное планирование и управление ресурсами, чтобы максимизировать эффективность разработки и минимизировать потери.

Эти принципы бережливого производства могут быть применены в контексте разработки ИС БСТС с целью повышения эффективности, качества и устойчивости процесса разработки, а также обеспечения соответствия требованиям пользователей и оптимальной работы в сельскохозяйственной среде.

3.3. Шаблоны проектирования в синтезе с бережливым производством

Для возможности применения указанных выше принципов необходима стандартизированная и универсальная система – шаблоны проектирования. Шаблоны проектирования, рассмотренные в источниках [28, 29], являются основным ключевым механизмом применения бережливого производства и может принести следующие преимущества:

1. Стандартизация процессов: шаблоны проектирования ИС предоставляют стандартные наборы компонентов, архитектурных решений и процессов, которые могут быть повторно использованы в различных проектах. Это позволяет достичь стандартизации процессов разработки и улучшить эффективность команды разработчиков. Кроме того, стандартизация позволяет сократить время разработки, упростить поддержку и обеспечить более предсказуемые результаты.

2. Ускорение разработки: шаблоны проектирования ИС содержат заранее определенные компоненты и модули, которые могут быть мгновенно

внедрены в новых проектах. Это позволяет сократить время, затрачиваемое на разработку и интеграцию базовой функциональности системы, так как необходимые элементы уже готовы к использованию. Быстрая разработка и внедрение системы способствуют сокращению времени до достижения целей проекта и позволяют быстрее получить обратную связь от пользователей.

3. Снижение рисков: шаблоны проектирования ИС основаны на проверенных и признанных практиках разработки. Это позволяет снизить риски, связанные с проектированием и разработкой системы с нуля. Использование проверенных шаблонов и компонентов повышает вероятность успешной реализации проекта и уменьшает возможность ошибок и проблем, которые могут возникнуть при разработке новой системы.

4. Улучшение качества и надежности: шаблоны проектирования ИС основаны проверенных и оптимальных решениях, которые были разработаны на основе опыта и знаний. Использование таких шаблонов способствует повышению качества и надежности системы, так как они уже прошли испытание временем и могут быть применены для различных сценариев использования.

5. Упрощение поддержки и обновлений: Использование шаблонов проектирования ИС упрощает процесс поддержки и обновлений системы. Поскольку шаблоны уже предоставляют структуру и архитектуру системы, поддержка и обновления могут быть более стандартизированными и прозрачными. Изменения и улучшения могут быть внесены в отдельные компоненты или модули, а не во всю систему целиком, что уменьшает риск возникновения ошибок и облегчает процесс тестирования и внедрения изменений.

6. Улучшение масштабируемости: шаблоны проектирования ИС обычно предусматривают гибкую и масштабируемую архитектуру, которая позволяет системе легко расширяться и адаптироваться к растущим требованиям и изменяющимся потребностям. Это обеспечивает возможность

быстрого масштабирования системы, без значительных изменений в ее основной архитектуре и функциональности.

7. Повышение эффективности разработчиков: шаблоны проектирования ИС предоставляют разработчикам готовые решения и компоненты, которые могут быть использованы в различных проектах. Это позволяет разработчикам сосредоточиться на более сложных и специфических задачах, вместо траты времени на повторное создание основных элементов системы. Повышение эффективности разработчиков способствует ускорению процесса разработки, улучшению качества кода и снижению затрат на разработку.

8. Обмен знаниями и опытом: шаблоны проектирования ИС являются не только готовыми решениями, но и инструментами для обмена знаниями и опытом между разработчиками. Разработчики могут изучать и анализировать шаблоны, чтобы понять проверенные подходы и лучшие практики в разработке информационных систем. Это способствует накоплению знаний в команде, повышению ее компетентности и возможности для дальнейшего улучшения процессов разработки.

Использование шаблонов информационных систем с точки зрения бережливого производства позволяет повысить эффективность разработки, сократить время и ресурсы, улучшить качество и надежность системы, а также облегчить поддержку и обновления. Стандартизированные и проверенные шаблоны предоставляют основу для быстрого и надежного создания ИС, снижая риски и повышая вероятность успешной реализации проекта.

В целом, бережливое производство и использование шаблонов информационных систем в разработке БСТС взаимосвязаны и взаимодополняют друг друга. Они обеспечивают систематический и эффективный подход к разработке, с упором на минимизацию потерь, повышение качества и непрерывное улучшение системы.

В свою очередь неиспользование шаблонов при проектировании ИС БСТС приводит к таким негативным последствиям, как:

1. Несогласованность и хаотичность - различные разработчики могут принимать разные подходы и решения, что затрудняет понимание и сопровождение системы.

2. Увеличение трудоемкости – разработчику придется заново решать типичные задачи и проблемы, что может быть очень трудоемким и затратным процессом. Это также увеличивает риск возникновения ошибок и неправильных решений.

3. Ограниченная переносимость и масштабируемость – в отсутствие шаблонов проектирования ИС БСТС могут быть сложными для переноса на другие платформы или масштабирования для удовлетворения возрастающих требований. Стандартизированные шаблоны помогают сделать систему более гибкой и адаптируемой.

4. Увеличение затрат – разработка ИС БСТС может требовать больше времени и ресурсов. Разработчики могут сталкиваться с проблемами, которые уже были решены в других проектах, но теперь приходится решать их снова, что приводит к дополнительным затратам.

3.4. Применение бережливого производства и шаблонов проектирования к ИС БСТС

Далее рассмотрим преимущества для каждой из подсистем ИС БСТС при использовании шаблонов проектирования в концепции бережливого производства:

Позиционирование и навигация

1. Высокая точность и надежность – шаблоны проектирования могут включать оптимизированные алгоритмы и методы для определения местоположения и точного времени. Это может включать использование фильтров Калмана для объединения данных с разных источников, компенсацию ошибок измерений, а также механизмы обнаружения и коррекции ошибок. Вследствие этого обеспечивается высокая точность и

надежность позиционирования, что особенно важно для безопасной и эффективной навигации БСТС в сельскохозяйственных условиях.

2. Использование стандартных протоколов и форматов данных – шаблоны проектирования могут предлагать стандартизированные протоколы и форматы данных для обмена информацией между различными компонентами системы позиционирования и навигации. Это облегчает интеграцию различных устройств и систем, а также обеспечивает совместимость с другими системами, такими как глобальные навигационные спутниковые системы (ГНСС).

3. Управление множеством датчиков – в системе позиционирования и навигации используются различные датчики, такие как глобальные навигационные спутниковые системы, инерциальные измерительные блоки и другие. Шаблоны проектирования могут предоставлять структурированные подходы к управлению и взаимодействию с множеством датчиков, обеспечивая синхронизацию данных и обработку информации с минимальными задержками.

4. Гибкость и адаптивность – шаблоны проектирования позволяют создавать гибкие и адаптивные системы позиционирования и навигации. Они предлагают модульные и расширяемые архитектуры, которые позволяют легко добавлять новые функциональности и адаптироваться к различным сельскохозяйственным условиям и требованиям.

5. Интеграция с другими подсистемами – шаблоны проектирования обеспечивают интеграцию подсистемы позиционирования и навигации с другими компонентами системы автоматизации БСТС. Это позволяет эффективно обмениваться информацией, координировать действия различных подсистем и обеспечивать согласованную работу всей системы.

Коммуникация и связь

1. Устойчивость системы – использование опробованных сочетаний и комбинаций стандартных протоколов передачи данных и коммуникации, таких как TCP/IP, GSM, Wi-Fi, Bluetooth и других. Кроме того, это позволяет

обеспечить совместимость и интеграцию с другими системами, а также облегчает обмен данными между компонентами системы коммуникации и связи.

2. Модульность и гибкость – позволяет заменять или модифицировать отдельные компоненты без влияния на работу всей системы. Это облегчает разработку, сопровождение и обновление системы, а также позволяет адаптировать ее к различным требованиям и условиям использования.

3. Обеспечение безопасности – использование готовых методов и механизмов, направленных на обеспечение безопасности системы коммуникации и связи. Это может включать использование шифрования данных, аутентификацию, контроль доступа и другие меры, чтобы защитить информацию от несанкционированного доступа или подделки.

4. Повышение помехоустойчивости и исправление ошибок – применение механизмов для обнаружения и исправления ошибок в системе коммуникации и связи. Это может включать использование протоколов с повторной передачей данных, обработку и восстановление после сбоев, резервирование и другие методы, которые позволяют системе эффективно справляться с возможными сбоями или потерей связи.

5. Оптимизация производительности – использование уже определенных в шаблоне алгоритмов сжатия данных, кэширование информации, распределение нагрузки и другие методы, направленные на улучшение скорости и эффективности передачи данных и обработки сигналов.

Машинное зрение

1. Экономия на квалификации исполнителей – разработчики могут использовать уже существующие шаблоны, которые были разработаны и протестированы опытными специалистами, чтобы создать систему машинного зрения.

2. Снижение трудоемкости – использование структурированных и оптимизированных решений для различных компонентов системы машинного

зрения, таких как обработка изображений, классификация, детектирование и другие. Использование шаблонов позволяет снизить трудоемкость разработки, поскольку разработчику не нужно заново изобретать каждый компонент системы.

3. Минимизация номенклатуры элементной базы – определено оптимальное расположение сенсорики, уже выбраны подходящие параметры и компоненты системы машинного зрения. Разработчики могут ориентироваться на рекомендации и проверенные решения, избегая необходимости тестирования и внедрения новых, неизвестных компонентов.

4. Ускорение процесса разработки – упрощает и ускоряет процесс разработки системы машинного зрения. Разработчикам не нужно проводить много времени на поиск и эксперименты с различными подходами, так как шаблоны предлагают готовые решения, которые можно легко адаптировать к конкретным потребностям системы.

5. Снижение рисков и ошибок – использование проверенных и надежных решений, которые были применены в реальных проектах. Использование шаблонов позволяет избежать возможных проблем, связанных с размещением сенсорики, выбором параметров и другими аспектами проектирования. Это снижает вероятность возникновения ошибок и повышает надежность системы.

Вычислительная подсистема БСТС

1. Эффективное управление ресурсами – шаблоны проектирования позволяют оптимизировать использование вычислительных ресурсов, таких как процессоры, память и сетевые ресурсы. Они предлагают структурированные подходы к управлению вычислениями, распределению задач и оптимизации производительности системы, что особенно важно для систем с ограниченными ресурсами, таких как ИС БСТС. Это позволяет вычислительной подсистеме эффективно выполнять сложные вычисления и обеспечивать высокую производительность ИС БСТС.

2. Автономное принятие решений – шаблоны проектирования могут включать алгоритмы и методы для автономного принятия решений на основе анализа данных, полученных от других подсистем БСТС. Это позволяет вычислительной подсистеме самостоятельно анализировать информацию, прогнозировать события и принимать решения в режиме реального времени. Такой подход повышает автономность и адаптивность БСТС.

3. Гибкость и масштабируемость – шаблоны проектирования предоставляют гибкую архитектуру для вычислительной подсистемы, которая позволяет легко добавлять новые функциональности, модифицировать существующие компоненты и масштабировать систему по мере необходимости. Это особенно важно в условиях развития технологий и изменения требований сельскохозяйственных задач.

4. Обработка больших объемов данных – шаблоны проектирования могут включать методы и структуры данных для эффективной обработки больших объемов данных, которые генерируются различной сенсорикой. Это позволяет вычислительной подсистеме обрабатывать, анализировать и извлекать полезную информацию из больших данных, что является важным аспектом в сельскохозяйственных задачах.

5. Надежность и безопасность – шаблоны проектирования позволяют уделять особое внимание вопросам безопасности и надежности вычислительной подсистемы. Они предлагают методы для обработки ошибок, обеспечения целостности данных и защиты от внешних угроз. Это позволяет БСТС функционировать безопасно и надежно в различных условиях сельскохозяйственной среды.

В целом, применение шаблонов проектирования в концепции бережливого производства при разработке ИС БСТС позволяет повысить эффективность, гибкость и надежность системы, сократить время и затраты на разработку, а также обеспечить соответствие требованиям и ожиданиям пользователей. Это способствует созданию инновационных и

конкурентоспособных решений для автоматизации сельскохозяйственных задач.

Детализация шаблонов проектирования ИС БСТС в рамках взаимоувязки бережливого производства и одной из основополагающих концепций повышения интегральной производительности инженерных систем в целом - теории ограничений Голдратта будет рассмотрена в следующей главе.

Заключение главы 1

В данной главе был проведен обзор существующих и перспективных БСТС в сельском хозяйстве; были рассмотрены инфокоммуникационные системы БСТС и методологии их проектирования.

В первом разделе была проведена классификация БСТС по уровням автономности, рассмотрены средства автоматизации и различные типы наземной сельскохозяйственной техники с функционалом БСТС. Также было произведено сравнение возможностей средств автоматизации от различных производителей и представлен обзор современных российских сельскохозяйственных БСТС.

Важным аспектом работы с БСТС являются сценарии их эксплуатации, которые были рассмотрены в данной главе. Были выделены требования и основные направления создания современных сельскохозяйственных БСТС, учитывая особенности работы в аграрной сфере.

Во втором разделе был рассмотрен состав и особенности систем автоматизации БСТС, выделены их основные компоненты, а также представлена структура инфокоммуникационных систем транспортного средства.

В третьей части были изучены методологии проектирования информационных систем БСТС. Были рассмотрены современные практики использования методологий проектирования и сформулированы основные положения для последующего внедрения бережливого производства в контексте разработки ИС БСТС. Особое внимание было уделено шаблонам проектирования и их интеграции с концепцией бережливого производства.

В заключении первой главы можно сделать вывод о том, что разработка и внедрение инфокоммуникационных систем в сельскохозяйственные транспортные средства является сложным и многогранным процессом. Использование методологии бережливого производства и шаблонов проектирования позволяет значительно повысить эффективность и качество

разработки, обеспечивая стандартизацию, повторное использование, упрощение поддержки и обновлений, снижение рисков и ошибок.

Дальнейшее исследование будет направлено на более детальное изучение применения бережливого производства, целеполагания по Голдратту и шаблонов проектирования в инфокоммуникационных системах БСТС, а также на разработку конкретных рекомендаций и руководств по их использованию для достижения оптимальных результатов в разработке и эксплуатации БСТС в сельском хозяйстве.

Глава 2 Реализация шаблонов проектирования в различных ИС БСТС с точки зрения реализации концепции бережливого производства и теории ограничения Голдратта

1. Введение в бережливое производство и методологию управления ограничениями

В этом разделе мы представим введение в концепцию бережливого производства, изучим историю, основные принципы и применение этой концепции в современных организациях. Затем мы перейдем к изучению теории ограничений, включая определение системного ограничения, принципы управления ограничениями и процесс управления ограничениями. Особое внимание будет уделено определению цели системы и значению глобальных показателей эффективности.

Мы также рассмотрим методологию управления ограничениями, включая ее применение в бережливом производстве и расширенную методологию. Представлены примеры применения методологии управления ограничениями для более наглядного понимания.

1.1 Введение в бережливое производство

1.1.1 История и развитие бережливого производства

В 1950-1960 годах в Toyota была разработана система производства, известная как Toyota Production System (TPS), которая стала предвестником бережливого производства. Основой TPS были принципы эффективного использования ресурсов, устранения потерь и непрерывного улучшения процессов. Эти принципы были внедрены в производственные операции Toyota и принесли компании значительный успех.

В 1990-х годах концепция бережливого производства была формализована и описана в книге [30]. Эта книга рассказывает об исследовании, проведенном Международным институтом менеджмента, Группой автомобильной промышленности и Массачусетским технологическим институтом, в котором были проанализированы

производственные системы автомобильной промышленности в разных странах. Она показала, что система управления производством Toyota является более эффективной и гибкой по сравнению с традиционными системами.

1.1.2 Основные принципы бережливого производства

Основные принципы бережливого производства основаны на идее эффективного использования ресурсов, устранения потерь и непрерывного улучшения [31]. Перечень основных принципов:

1. Создание ценности для клиента: бережливое производство ставит фокус на потребности и ценность для клиента. Основная цель - предоставить клиенту продукт или услугу с высоким качеством, в нужное время и по приемлемой цене.

2. Устранение потерь: бережливое производство стремится к устранению всех видов потерь в процессе производства или предоставления услуги. Потери могут включать переработку, ожидание, перенос, избыточные запасы и т. д. Цель состоит в том, чтобы достичь максимальной эффективности и минимизировать ресурсозатраты.

3. Постоянное улучшение: бережливое производство подразумевает постоянное совершенствование процессов и систем. Компании, применяющие бережливое производство, стремятся к непрерывному улучшению качества, производительности и эффективности.

4. Поддержка сотрудников: бережливое производство уделяет внимание вовлеченности сотрудников и поддержке их развития. Он признает важность командной работы, обучения и развития навыков для достижения целей организации.

1.1.3 Применение бережливого производства в современных организациях

Бережливое производство нашло широкое применение в различных отраслях и организациях, в том числе и в агропромышленном секторе [32].

Независимо от того, производственная это компания или организация, предоставляющая услуги, бережливое производство может помочь улучшить эффективность операций и достичь высокого уровня клиентской удовлетворенности.

Множество компаний по всему миру внедряют бережливое производство в свои бизнес-процессы. Они применяют инструменты и методы, такие как Kaizen[33], 5S[34], Value Stream Mapping[35] и многие другие, чтобы улучшить производительность, качество и доставку продуктов и услуг.

Применение бережливого производства также помогает организациям стать более гибкими и отзывчивыми на изменения на рынке и потребности клиентов. Оно способствует повышению конкурентоспособности и устойчивости организации в динамичной бизнес-среде.

Таким образом, применение бережливого производства становится неотъемлемой частью стратегии развития организаций и способствует их росту и успеху на рынке.

1.2 Теория ограничений: основные понятия и принципы

1.2.1 Введение в теорию ограничений

Теория ограничений (Theory of Constraints, TOC) – это системный подход к управлению и улучшению операций, который был разработан Исраэлем Голдраттом в 1980-х годах[36]. Она основывается на предположении о том, что любая система имеет по крайней мере одно ограничение, которое препятствует ее эффективности и производительности.

1.2.2 Определение системного ограничения

Системное ограничение – это элемент или фактор, который ограничивает или замедляет работу системы в целом. Оно является узким местом или слабым звеном в процессе, которое влияет на общую производительность системы. Определение системного ограничения является важным шагом в теории ограничений, поскольку позволяет

сконцентрироваться на ключевых областях для улучшения производительности системы[37].

1.2.3 Принципы управления ограничениями

Управление ограничениями в теории ограничений базируется на нескольких основных принципах:

- Операционное управление: фокусировка на управлении ограничением системы для достижения максимального результата. Это включает определение и управление пропускной способностью ограничения, контроль потока материалов и информации, и управление применением ресурсов.
- Синхронизация: согласованность и согласованность операций в системе, чтобы избежать ненужных запасов, задержек и потерь времени. Синхронизация включает в себя выравнивание процессов, устранение узких мест и оптимизацию потока работы.
- Подчинение всего остального решению: при принятии решений необходимо учитывать влияние на системное ограничение. Подчинение всего остального решению означает, что все решения и действия должны быть нацелены на максимальное использование и эффективное управление ограничением.
- Постоянное улучшение: применять принцип постоянного улучшения для достижения прогресса и развития системы. Это включает поиск новых возможностей, инноваций и улучшений процессов для достижения более высокой производительности и эффективности.

1.2.4 Процесс управления ограничениями: шаги и методология

Процесс управления ограничениями включает ряд шагов и методологий, которые помогают систематически определить, управлять и улучшать системное ограничение. Основные шаги включают:

1. Определение системного ограничения: определите ключевой элемент или фактор, который ограничивает производительность системы в целом. Это

может быть физическое ограничение, ресурсное ограничение или ограничение в рабочих процессах.

2. Эксплуатация системного ограничения: разработайте стратегию и методы, чтобы максимально использовать ограничение и увеличить его пропускную способность. Это может включать оптимизацию рабочих методов, использование специализированного оборудования или управление временем.

3. Подчинение всего остального решению: установите контрольные механизмы и процессы, чтобы подчинить все решения и действия потребностям системного ограничения. Это позволяет избежать перегрузки ограничения и оптимизировать использование ресурсов.

4. Повышение системного ограничения: разработайте стратегии и методы для повышения пропускной способности системного ограничения. Это может включать внедрение новых технологий, улучшение процессов или расширение ресурсов.

5. Циклический процесс: процесс управления ограничениями является циклическим и требует постоянного наблюдения, анализа и улучшения. Если в любом из предыдущих шагов ограничение нарушается или изменяется, процесс возвращается к шагу 1 для переоценки и обновления стратегии управления ограничением.

Теория ограничений и ее методология обеспечивают систематический подход к управлению и улучшению производительности организации. Разработка эффективной стратегии управления ограничениями и применение шагов и методологий позволяют достичь более высокой производительности, эффективности и конкурентоспособности.

1.2.5 Расширенная методология управления ограничениями

Расширенная методология управления ограничениями, основанная на теории ограничений, представляет собой более полный и комплексный подход к управлению ограничениями. Она включает не только выявление и эксплуатацию системного ограничения, но и определение системной цели,

определение глобальных показателей эффективности и проведение анализа цепей ограничений.

Расширенная методология также включает в себя шаги по подчинению всего остального решению системному ограничению и постоянному повышению системного ограничения. Она предоставляет организации более полное представление о ее производительности и помогает сосредоточить усилия на улучшении ключевых аспектов, способствующих достижению системных целей.

1.2.6 Примеры применения методологии управления ограничениями

Методология управления ограничениями широко применяется в различных отраслях и организациях. Некоторые примеры применения этой методологии включают:

- В производственной отрасли: методология управления ограничениями может быть использована для определения и устранения «бутылочных горлышек» в производственных процессах, повышения производительности и сокращения времени цикла производства.
- В сфере услуг: методология управления ограничениями может помочь организациям в сфере услуг определить главное ограничение, улучшить процессы обслуживания клиентов и повысить удовлетворенность клиентов.
- В проектном управлении: методология управления ограничениями может быть использована для определения критического пути в проекте, управления ресурсами и сроками выполнения задач, а также для повышения эффективности проектных команд.

Приведенные примеры демонстрируют широкий спектр применения методологии управления ограничениями и ее важность в достижении эффективности и конкурентоспособности в различных организациях и отраслях.

2. Методология управления ограничениями в ИС БСТС

Методология управления ограничениями является важной составляющей бережливого производства и позволяет организациям достигать оптимальной производительности и эффективности. Она включает в себя использование принципов и инструментов теории ограничений для выявления и управления ограничениями в системе.

В рамках бережливого производства методология управления ограничениями применяется для постоянного улучшения процессов и устранения «бутылочных горлышек». Это достигается путем идентификации главного ограничения, разработки стратегий его эксплуатации и выравнивания всего остального в системе в соответствии с этим ограничением. Далее определим главные ограничения для каждой из подсистем ИС БСТС.

2.1 Определение системных ограничений в ИС БСТС

Определение ограничений для каждой из подсистем происходит в несколько этапов, начиная с анализа требований к системе и заканчивая анализом рисков и сценариев использования. Процесс выбора ограничений включает следующие шаги:

1. **Анализ требований**: в начале исследования были проанализированы требования к системе, включая функциональные, технические, производственные и экономические аспекты. Это позволило определить основные цели и задачи системы, а также выявить потенциальные риски и проблемные области.

2. **Идентификация возможных ограничений**: на основе анализа требований выделяются потенциальные ограничения, которые могут влиять на функционирование системы или ее способность удовлетворять поставленные требования. Это могут быть ограничения как технического характера (например, ограничения по скорости или точности), так и

ограничения, связанные с ресурсами (например, ограничения по энергопотреблению или вычислительным ресурсам).

3. **Анализ рисков и сценариев использования**: для каждой подсистемы был проведен анализ рисков и сценариев использования, что позволило выявить потенциальные угрозы и опасности, а также определить критические сценарии, в которых система может столкнуться с наибольшими трудностями или ограничениями.

4. **Выбор конечных ограничений**: на основе анализа требований, идентификации возможных ограничений и анализа рисков выбираются конечные ограничения для каждой подсистемы. Этот процесс включает в себя оценку потенциальных рисков и проблем, а также учет технических и экономических возможностей.

Таким образом, процесс выбора ограничений основан на комплексном анализе требований, рисков и сценариев использования, что позволит определить наиболее критические и значимые ограничения для каждой из подсистем.

2.1.1 Определение системного ограничения в вычислительной подсистеме ИС БСТС

В вычислительной подсистеме ИС БСТС существуют два важных системных ограничения: ограниченность ресурсов и строгая приоритезация задач.

1. **Ограниченность ресурсов** вычислительной подсистемы представляет собой ограниченное количество доступных ресурсов, таких как процессорное время, память и пропускная способность. Беспилотная система требует значительных вычислительных ресурсов для обработки больших объемов данных, выполнения сложных алгоритмов и принятия решений в реальном времени. В силу ограниченности этих ресурсов, вычислительная подсистема должна эффективно управлять ими и распределять их между задачами, выполняемыми в рамках ИС БСТС. Неэффективное использование ресурсов может привести к задержкам в обработке данных, увеличению

времени отклика системы, низкой производительности, ограниченной возможности масштабирования системы и неправильному функционированию. Поэтому оптимальное использование ресурсов является ключевым аспектом разработки вычислительной подсистемы и обеспечивает ее стабильную и надежную работу.

2. Другим важным ограничением является **строгая приоритезация задач** внутри ИС БСТС. Это означает, что задачи, выполняемые в рамках системы, должны быть правильно организованы по приоритетам. Например, при детектировании препятствий или других аварийных ситуациях, задачи, связанные с реакцией на препятствие и корректировкой маршрута, должны получить наивысший приоритет. Это обеспечивает безопасность и предотвращает возможные аварии. Кроме того, внешние факторы, такие как погодные условия и сезонные работы, могут также влиять на приоритетность определенных операций в сельскохозяйственном процессе. Например, при наличии неблагоприятных погодных условий, задачи, связанные с защитой урожая или обработкой почвы, могут получить приоритет перед другими задачами. Это позволяет эффективно использовать ресурсы и временные окна, способствуя повышению производительности и снижению потенциальных убытков.

Таким образом, ограниченность ресурсов вычислительной подсистемы и строгая приоритезация задач в ИС БСТС являются важными системными ограничениями. Управление этими ограничениями требует эффективного распределения ресурсов, определения приоритетов задач и поддержания стабильной работы системы в условиях ограниченных ресурсов и меняющихся внешних факторов.

2.1.2 Определение системного ограничения в подсистеме коммуникации и связи ИС БСТС

При проектировании подсистемы коммуникации и связи ИС БСТС существуют несколько системных ограничений, которые необходимо учитывать:

1. **Пропускная способность каналов связи**. Пропускная способность каналов связи ограничивает объем данных, который система может передавать в определенный момент времени. В контексте БСТС, где сенсоры, управляющие блоки и центральное управление постоянно обмениваются данными, недостаточная пропускная способность может вызвать задержки в передаче данных. Это, в свою очередь, может привести к потере важной информации, а также к снижению реакции системы на изменения в окружающей среде, что является недопустимым в автономных системах.

2. **Латентность и задержки**. Латентность (задержки) в передаче данных ограничивает скорость обмена информацией между компонентами системы. В БСТС требуется быстрая реакция на изменяющиеся условия дорожного движения, окружающей среды и команды от центрального управления. Даже небольшие задержки могут существенно повлиять на безопасность и эффективность работы системы, а также могут увеличить риск возникновения аварийных ситуаций.

3. **Помехоустойчивость**. Система должна быть устойчивой к возможным помехам и интерференциям в каналах связи. Это важно для обеспечения надежной передачи данных даже в условиях неблагоприятных окружающих факторов. В условиях реального мира существует множество потенциальных источников помех и интерференций в каналах связи, такие как электромагнитные помехи, метеорологические условия и другие. Эти помехи могут нарушить передачу данных и вызвать ошибки в интерпретации информации. Для надежной работы системы в любых условиях необходима высокая степень помехоустойчивости, чтобы минимизировать вероятность ошибок и потери данных.

4. **Защита данных.** Защита данных становится особенно важной, когда речь идет о сельскохозяйственной информации или управляющих командах, которые могут иметь стратегическое значение. Несанкционированный доступ к такой информации может привести к серьезным последствиям, включая утерю конфиденциальности и возможные атаки на систему. Следовательно,

необходимо обеспечивать высокий уровень секретности и целостности данных при их передаче и хранении.

5. **Совместимость с разными устройствами.** Совместимость является ключевым аспектом при проектировании системы коммуникации и связи, так как различные устройства и протоколы связи могут использоваться в разных компонентах БСТС. Отсутствие совместимости может привести к сложностям в интеграции нового оборудования или технологий в систему, что может потребовать дорогостоящих изменений в структуре системы и снизить гибкость системы в адаптации к новым условиям.

2.1.3 Определение системного ограничения в подсистеме позиционирования и навигации ИС БСТС

При разработке подсистемы позиционирования и навигации в ИС БСТС, следует учитывать следующие системные ограничения:

1. **Точность позиционирования**: недостаточная точность определения местоположения транспортных средств и оборудования является критическим ограничением, так как она может привести к навигационным ошибкам и аварийным ситуациям. Высокая точность позиционирования становится важной, особенно в контексте автономных систем, где безопасность и эффективность зависят от правильных координат.

2. **Доступность сигналов навигационных спутников**: ограничение доступности сигналов навигационных спутников может возникнуть в ситуациях, таких как в ущельях или городских каньонах, где прямая видимость к спутникам ограничена. Это создает угрозу для навигации, и разработчики должны предусмотреть механизмы для обеспечения надежного позиционирования даже в условиях ограниченной видимости спутников.

3. **Зависимость от глобальных систем навигации**: многие современные системы позиционирования и навигации зависят от глобальных систем навигации, таких как GPS или ГЛОНАСС. Сбои или недоступность этих систем могут серьезно нарушить работу подсистемы, поэтому необходимо рассматривать альтернативные методы позиционирования и

резервное хранилище данных для обеспечения надежности и устойчивости навигации.

2.1.4 Определение системного ограничения в подсистеме машинного зрения ИС БСТС

При проектировании подсистемы машинного зрения в ИС БСТС для использования в сельском хозяйстве, необходимо учитывать ряд системных ограничений. Эти ограничения обусловлены спецификой задач, выполняемых в сельском хозяйстве, а также особенностями работы в полевых условиях. В данном контексте подсистема машинного зрения является ключевой для успешной реализации сельскохозяйственных операций, таких как посев, уборка урожая, контроль состояния растений и многие другие. В этой связи представляется важным детально рассмотреть и обосновать системные ограничения данной подсистемы.

1. **Качество обработки изображений.** Сельскохозяйственные операции требуют высокой точности в распознавании и анализе объектов, таких как растения, сельскохозяйственная техника и сельскохозяйственные участки. Недостаточное качество обработки изображений может привести к неправильному распознаванию объектов, что, в свою очередь, может вызвать аварии и несчастные случаи. Высокое качество обработки обеспечивает точность и надежность в выполнении сельскохозяйственных задач.

2. **Эффективность в условиях неблагоприятных факторов.** Сельское хозяйство подразумевает работу в разнообразных климатических и окружающих условиях. Подсистема машинного зрения должна быть спроектирована с учетом снижения воздействия неблагоприятных факторов, таких как плохая видимость из-за атмосферных условий, изменения освещения или наличие загрязнений на камерах и объективах. Неправильное функционирование в таких условиях может существенно ограничить способность системы выполнять сельскохозяйственные задачи.

3. **Скорость обработки данных.** Сельскохозяйственные операции часто требуют быстрого анализа и принятия решений на основе данных с камер и

датчиков. Системное ограничение в виде недостаточной скорости обработки данных может привести к задержкам в реакции на изменяющиеся условия, что нежелательно в сельскохозяйственных операциях, где необходима оперативность и точность.

4. **Распознавание разнообразных объектов.** В сельском хозяйстве существует множество разных объектов, которые необходимо распознавать и анализировать, включая различные сорта растений, состояние почвы, наличие болезней и вредителей, а также сельскохозяйственную технику. Если система машинного зрения ограничена в распознавании разнообразных объектов, это может снизить ее полезность и применимость в сельскохозяйственных операциях.

2.2 Снятие системных ограничений в ИС БСТС

2.2.1 Снятие системного ограничения в вычислительной подсистеме ИС БСТС

Ранее было установлено, что в вычислительной подсистеме ИС БСТС существуют ограниченные ресурсы и строгая приоритезация задач. Предположение о том, что использование дополнительных средств или аппаратного обеспечения с большими ресурсами может решить эту проблему, оказывается несостоятельным. Фактически, такой подход только ухудшает систему в целом, приводя к увеличению стоимости, увеличению размеров разрабатываемого оборудования, сложностей в его компоновке и размещении на борту БСТС и т.д.

Снятие ограниченности ресурсов

Для преодоления ограниченности ресурсов в вычислительной подсистеме ИС БСТС предлагается использовать трехуровневую схему (Рисунок 2.1):

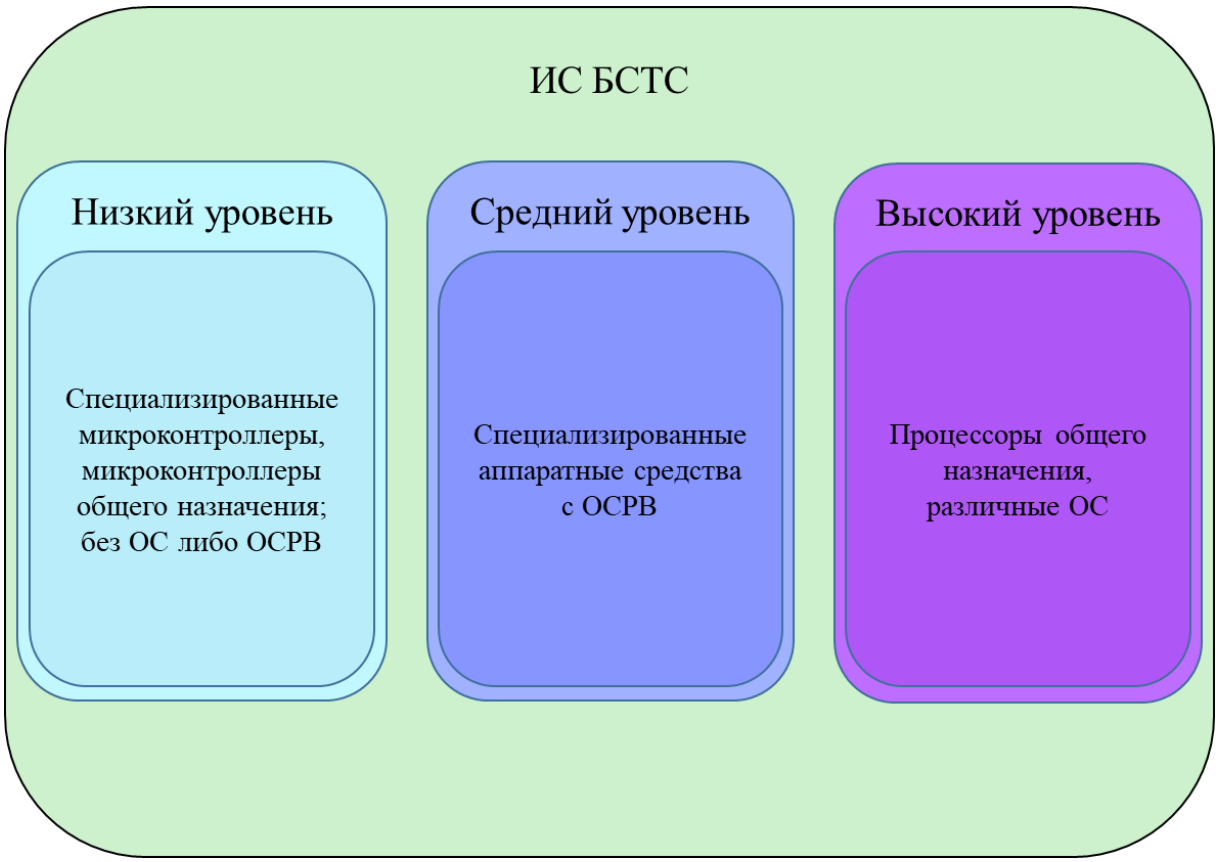


Рисунок 2.1 – трехуровневая схема ИС БСТС

Каждый уровень выполняет определенные задачи и использует соответствующие аппаратные и программные ресурсы:

1. **Низкий уровень.** На низком уровне системы используются специализированные микроконтроллеры, которые функционируют без операционной системы или на базе операционных систем жесткого реального времени (например, FreeRTOS[38]). Эти микроконтроллеры обеспечивают выполнение задач с высокой точностью временных характеристик и низкой латентностью. Они предназначены для выполнения простых и быстрых операций, таких как считывание данных с датчиков, управление актуаторами и другие рутинные функции.

В качестве примера рассмотрим систему измерения угла поворота БСТС на базе резистивного датчика (Рисунок 2.2), отметим основные схемы реализации и их особенности.

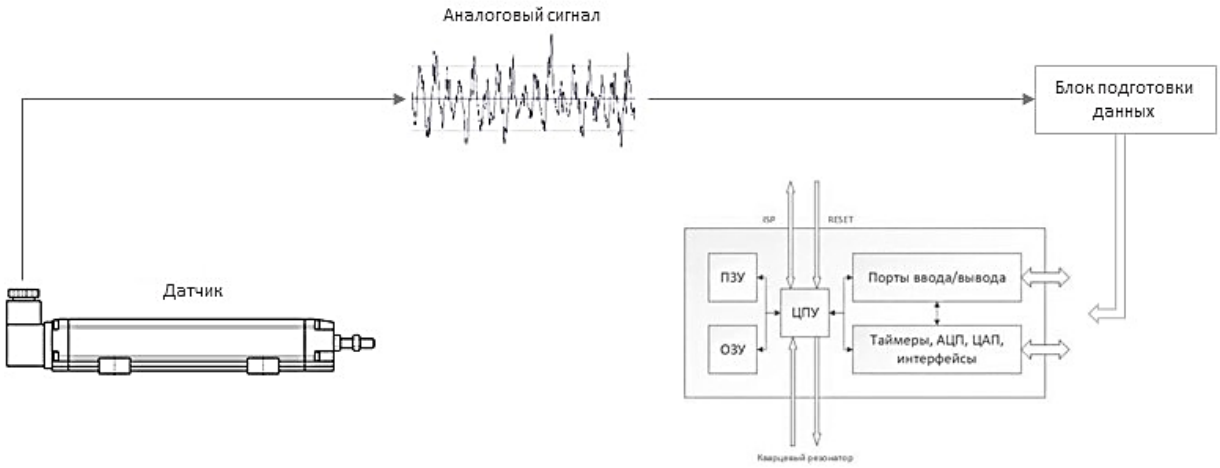


Рисунок 2.2 - Блок-схема измерительной системы

В качестве базового элемента измерительной системы может быть выбран микроконтроллер ARM архитектуры. К дополнительному оборудованию стоит отнести:

- датчик физических параметров;

- блок подготовки аналоговых сигналов (усиление, фильтрация и т.д.);

- периферийные средства ввода-вывода аналоговых сигналов (модули АЦП и ЦАП);

При разработке программного обеспечения используются следующие принципы: модульность, использование объектной метафоры в управлении, унификация связей и разделение программ управления.

Существует несколько методов реализации измерительной системы:

Схема “самых последних данных”. В этом методе реализации интерфейса АЦП работает непрерывно. В конце каждого цикла преобразования он обновляет данные в выходном буферном регистре и затем автоматически начинает новый цикл преобразования. Микропроцессор просто считывает содержимое этого буфера, когда ему нужны самые последние данные. Этот метод подходит для тех применений, где необходимость в обновлении данных возникает лишь от случая к случаю.

Схема “запуска-ожидания”. Микропроцессор инициирует выполнение преобразования каждый раз, когда ему нужны новые данные, и затем

непрерывно тестирует состояние АЦП, чтобы узнать, закончилось ли преобразование. Зафиксировав конец преобразования, он считывает выходное слово преобразователя. В возможной модификации этого метода микропроцессор находится в состоянии ожидания в течение интервала времени, превышающего предполагаемое время преобразования, и затем считывает выходные данные. Этот метод несколько проще в реализации, но при этом микропроцессор отвлекается от выполнения всех других программ на время преобразования.

Использование прерывания микропроцессора. Этот метод основан на возможности использования системы прерываний микропроцессора. Как и в предыдущей схеме, процессор или таймер запускают преобразователь, но затем микропроцессор может продолжать выполнение других заданий. Когда преобразование завершено, АЦП вызывает прерывание микропроцессора. Микропроцессор прекращает выполнение текущей программы и сохраняет всю необходимую информацию для последующего восстановления этой программы. Затем он осуществляет поиск и использование ряда команд (обслуживающая программа – обработчик прерывания), предназначенных для выборки данных от АЦП. После того как обслуживающая программа выполнена, микропроцессор возвращается к выполнению исходной программы.

Задача поиска обслуживающей программы иногда решается путем выполнения другой программы (программы или процедуры последовательного опроса – поллинга), которая определяет источник прерывания путем последовательной проверки всех возможных источников. Гораздо эффективнее подход, связанный с использованием векторных прерываний. Этот подход основан на хранении адресов отдельных обслуживающих программ в заранее определенной области памяти, называемой векторной таблицей. В ответ на сигнал прерывания микропроцессор теперь обращается к определенной ячейке памяти, в которую пользователем занесен адрес соответствующей обслуживающей программы.

Передача данных между АЦП и микропроцессором на программном уровне может быть организована тремя способами:

Передача через пространство основной памяти. При распределении памяти АЦП присваивается некоторый адрес в пространстве основной памяти, не используемый для фактического хранения данных и программ. Передача данных между АЦП и микропроцессором осуществляется путем обращения к АЦП просто как к ячейке памяти с данным адресом. Однако помимо уменьшения полезного пространства памяти такой подход может привести к усложнению управления памятью и, как правило, требует использования дополнительных аппаратных средств дешифрации адреса, поскольку при минимуме этих средств, слишком расточительно используется память.

Передача через пространство подсистемы ввода – вывода (ВВ). В некоторых системах создается отдельный набор адресов для подсистемы ВВ (пространство ВВ), которые могут совпадать по численным значениям с адресами ячеек основной памяти, но отличаются от них с помощью использования специальных управляющих сигналов, выдаваемых на системную шину. Отделение пространства памяти от пространства ВВ улучшает характеристики системы. Как правило, это позволяет довольно просто осуществлять дешифрацию адреса с использованием минимального количества аппаратных средств, поскольку “приносится в жертву” пространство ВВ, а не очень ценное пространство основной памяти.

Прямой доступ к памяти (ПДП). Если возникает необходимость только в простой передаче данных между памятью и каким-либо периферийным устройством, включение в интерфейс регистра - аккумулятора микропроцессора неоправданно уменьшает скорость передачи данных. Используя дополнительные аппаратные средства, обычно в виде специального устройства, называемого контроллером ПДП, можно осуществлять непосредственную передачу данных с гораздо большей скоростью. Большинство микропроцессоров допускает реализацию ПДП путем передачи управления системной шиной на определенный промежуток времени

контроллеру ПДП. Контроллер ПДП в течение этого промежутка времени управляет работой шины (захватывает шину) и обеспечивает передачу данных путем генерации соответствующих адресов и управляющих сигналов. Затем управление системной шиной передается обратно микропроцессору. Для передачи всех данных может потребоваться несколько таких ПДП-циклов. ПДП эффективен в тех применениях, где нужно обеспечить высокую скорость передачи данных или нужно передавать большие объемы данных. Применение этого метода в системах сбора данных в принципе возможно, но характерно только для систем с высокими рабочими параметрами.

2. **Средний уровень**. На среднем уровне системы осуществляется выполнение сложных задач, ограниченных по времени. Например, это могут быть задачи обработки изображений с помощью алгоритмов машинного зрения или принятие решений на основе анализа данных. Для реализации таких задач используются специализированные аппаратные средства, например, nVIDIA Xavier NX, которые обладают высокой производительностью и способны обрабатывать большие объемы данных[39]. В качестве операционной системы может быть использован Linux с расширением реального времени, таким как Linux deadline scheduler, который обеспечивает строгие гарантии временных характеристик выполнения задач.

В качестве примера рассмотрим систему машинного зрения на базе изделия nVIDIA Xavier NX (Рисунок 2.3):

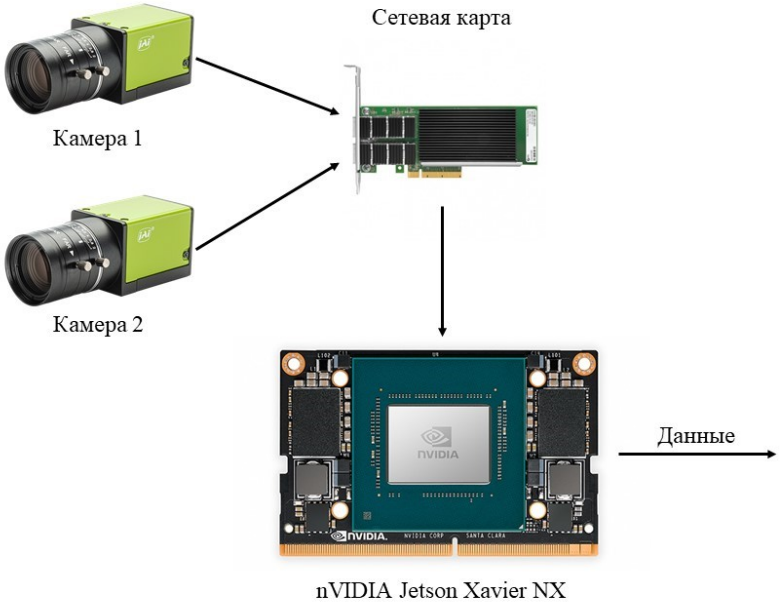


Рисунок 2.3 – блок схема системы машинного зрения

Запуск приложений на данном изделии предполагает работу ресурсоемких частей программы на GPU. В связи с этим требованием алгоритмы машинного зрения разрабатывались с помощью архитектуры параллельных вычислений CUDA. CUDA - программно-аппаратная архитектура параллельных вычислений от NVIDIA, которая позволяет увеличить скорость работы программы, благодаря использованию вычислительных микромодулей.

Общая схема системы машинного зрения состоит из следующих модулей:

- модуль чтения данных
- модуль препроцессинга изображений
- модуль распознавания объектов
- модуль нейронной сети

Модуль чтения данных принимает видеопоток с камеры и записывает последовательность изображений в оперативную память компьютера.

После этого модуль препроцессинга изображений забирает изображения из разделяемой памяти, обрабатывает и также записывает их в разделяемую память по другому адресу. После этого модуль распознавания объектов и модуль нейросети могут забирать обработанные с помощью препроцессинга картинки и детектировать на них объекты.

Модули нейросети и распознавания объектов предназначены для нахождения и классификации препятствий и других объектов на пути маршрута БСТС. Данные программные обеспечения являются демонами и не имеют графического интерфейса. В связи с этой особенностью данные программы работают в фоновом режиме. Запускаются и завершаются данные программы при запуске операционной системы. Программы находятся в постоянном ожидании входных картинок из разделяемой памяти, в случае если разделяемая память не создана, программы ждут ее создания. Разделяемая память выбрана для ускорения передачи изображения между различными программами на одной вычислительной машине. Данный вид IPC взаимодействия позволяет быстро передавать большие объемы данных между разными процессами, что и необходимо при создании системы реального времени. Однако при таком подходе необходимо защищать область общей памяти, чтобы избежать одновременного обращения к ней.

3. **Высокий уровень.** На высоком уровне системы выполняются разнообразные задачи, включая стратегическое планирование, обработку данных в большом объеме и задачи, которые не являются критичными по времени исполнения. Для выполнения таких задач на высоком уровне могут использоваться процессоры общего назначения, которые обеспечивают высокую вычислительную мощность и поддерживают различные операционные системы, включая ОС мягкого реального времени. Применение таких процессоров и операционных систем на высоком уровне позволяет эффективно обрабатывать задачи, требующие большого объема вычислительных ресурсов, а также выполнять задачи, которые не являются временно-критичными.

Например, в рамках системы БСТС на высоком уровне выполняется задача стратегического планирования маршрута для беспилотных сельскохозяйственных транспортных средств. Для этой задачи используются процессоры общего назначения, такие как Intel Core i7, и операционная система мягкого реального времени, например, Linux с патчем PREEMPT-RT.

При планировании маршрута система БСТС учитывает различные параметры, такие как топографию участка, наличие препятствий и требования к эффективности и экономичности работы. Процесс планирования маршрута включает в себя анализ и обработку большого объема данных, таких как карты, датчиковый поток информации и т.д.

Например, система получает данные о текущих условиях участка поля, включая информацию о почвенной влажности, рельефе и наличии препятствий. Используя алгоритмы и модели, система проводит анализ данных и определяет оптимальный маршрут для беспилотного транспортного средства. При этом учитываются различные факторы, такие как минимизация времени работы, оптимальное использование ресурсов и повышение точности и эффективности операций на поле.

Процесс планирования маршрута может быть разделен на несколько этапов, включающих сбор и предварительную обработку данных, применение алгоритмов планирования и генерацию оптимального маршрута. При этом процессоры общего назначения с высокой вычислительной мощностью и операционная система мягкого реального времени обеспечивают быстрое и эффективное выполнение вычислений, что позволяет системе реагировать на изменения внешних условий и генерировать актуальные маршруты для беспилотного транспортного средства. Пример реализации представлен на Рисунке 2.4:

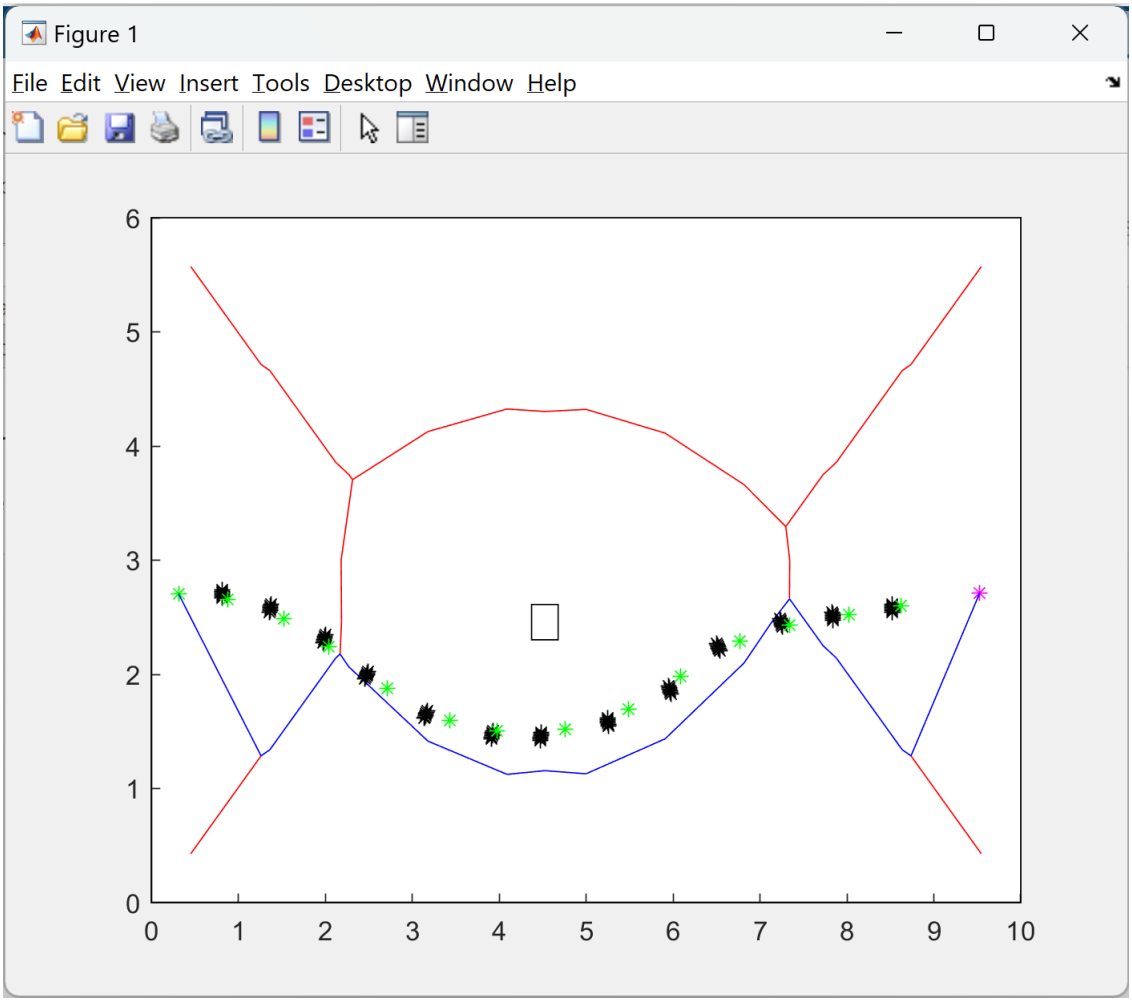


Рисунок 2.4 – Расчет в среде MATLAB траектории объезда препятствия (зеленые звездочки), отрисовка в среде MATLAB фактически пройденной трактором траектории (черные точки)

Таким образом, на высоком уровне системы БСТС применение процессоров общего назначения и операционных систем мягкого реального времени позволяет эффективно выполнять задачи стратегического планирования маршрута, обеспечивая оптимальное использование ресурсов и повышение эффективности операций в сельскохозяйственном процессе.

Снятие строгой приоритезации задач

Оптимальным и основным подходом для преодоления этих ограничений является использование систем реального времени. При проектировании и

разработке ИС БСТС следует учитывать особенности работы в реальном времени, чтобы обеспечить оптимальное выполнение задач с учетом их приоритетности.

Системой реального времени (СРВ) (real time operating systems (RTOS)) называется система, в которой успешность работы любой программы зависит не только от ее логической правильности, но и от времени, за которое она получила результат. Если временные ограничения не удовлетворены, то фиксируется сбой в работе системы.

Таким образом, временные ограничения должны быть гарантированно удовлетворены. Это требует от системы быть предсказуемой, то есть вне зависимости от своего текущего состояния и загруженности выдавать нужный результат за требуемое время. При этом желательно, чтобы система обеспечивала как можно больший процент использования имеющихся ресурсов.

Это предъявляет к системе вполне определенные базовые требования[40]. Рассмотрим требования, предъявляемые к системам реального времени:

Своевременная реакция. После того как произошло событие, реакция должна последовать не позднее, чем через требуемое время. Превышение этого времени рассматривается как серьезная ошибка.

Одновременная обработка информации, которая характеризует изменение процесса нескольких событий. Даже если одновременно происходит несколько событий, реакция ни на одно из них не должна запаздывать. Это означает, что система реального времени должна иметь встроенный параллелизм. Параллелизм достигается использованием нескольких процессоров в системе и/или многозадачным подходом.

Помимо прочего, различают сильное (hard) и слабое (soft) требование реального времени. Если запаздывание программы приводит к полному нарушению работы управляемой системы, то говорят о сильном реальном времени (жесткие СРВ). Если же запаздывание ведет только к потере

производительности, то говорят о слабом реальном времени (мягкие СРВ). Большинство программного обеспечения ориентировано на слабое реальное время, а задача хорошей СРВ - обеспечить уровень безопасного функционирования системы, даже если управляющая программа никогда не закончит своей работы.

Рассмотрим основные признаки систем жесткого и мягкого реального времени.

Признаки систем жесткого реального времени:

- недопустимость никаких задержек, ни при каких условиях;
- бесполезность результатов при опоздании;
- катастрофа при задержке реакции;
- цена опоздания бесконечно велика.

Пример системы жесткого реального времени - бортовая система управления самолетом.

Признаки систем мягкого реального времени:

- задержка реакции не критична;
- снижение производительности системы, вызванное запаздыванием реакции на происходящие события.

Пример - подсистема сетевого интерфейса. Если система не успела обработать очередной принятый пакет, можно восстановить его, используя сетевой протокол, повторяющий передачу пропущенных пакетов. При этом, конечно, произойдет снижение производительности системы.

Определим операционную систему реального времени (ОСРВ) как операционную систему, с помощью которой можно построить систему жесткого реального времени. **Обязательные требования** к ОСРВ:

Требование 1: ОСРВ должна быть многозадачной и поддерживать диспетчеризацию с вытеснением.

Это требование состоит в том, чтобы такая ОС была многонитиевой или многозадачной и, кроме того, планировщик должен иметь возможность вытеснять любую нить (задачу) и передавать управление той нити (задаче),

которая больше всего в этом нуждается. Для обеспечения вытеснения на уровне прерываний структура обслуживания прерываний (в том числе и аппаратная архитектура) должна быть многоуровневой.

Например, в ИС БСТС используется операционная система реального времени, которая способна одновременно выполнять несколько задач, связанных с управлением и контролем различных аспектов сельскохозяйственного транспорта - одна задача отвечает за мониторинг датчиков и обработку полученных данных, в то время как другая задача управляет движением транспортного средства. ОСРВ обеспечивает диспетчеризацию задач и их вытеснение в соответствии с приоритетами.

Требование 2: Должно существовать понятие приоритета нити (задачи).

Как найти нить (задачу), которая нуждается в ресурсах больше всего? В идеальном случае ОСРВ предоставляет ресурсы той задаче или драйверу, у которых осталось меньше всего времени до истечения срока реакции на событие (назовем такую ОС – ОС управляющей критическими сроками). Однако для реализации этого механизма нужно уметь прогнозировать, сколько времени понадобится задаче для завершения своей работы и сколько времени понадобится другим задачам для того, чтобы они успели к своим критическим срокам. Подобная ОСРВ пока еще не создана из-за сложности реализации. Поэтому разработчики ОС используют другой метод: они вводят концепцию приоритетов для нитей (задач).

При построении конкретной системы реального времени разработчик должен выстроить приоритеты задач таким образом, чтобы каждая из них успела с реакцией к своему критическому сроку, то есть он должен трансформировать базовое требование реального времени "успеть с реакцией к нужному моменту" в комбинацию приоритетов и в сценарий их динамического изменения. Очевидно, что при этой трансформации возможны ошибки, приводящие к неправильной работе системы. Для решения этого вопроса используют различные теории, такие как, теория монотонного планирования или различные методы и средства моделирования. Однако, эти

методы оказываются не всегда эффективными. Как бы то ни было, во всех современных ОСРВ приходится использовать механизм приоритетов как один из инструментов предсказуемости поведения системы. На сегодняшний день не имеется другого решения, понятие приоритета потока для систем реального времени неизбежно.

На примере БСТС приоритеты задач могут быть установлены в зависимости от их критичности и важности для безопасности и эффективности работы – задача реакции на препятствие может иметь более высокий приоритет, чем задача следования по запланированному маршруту. Это позволяет обеспечить надлежащую реакцию системы на внешние события и сохранить безопасность операций.

Требование 3: ОС должна поддерживать предсказуемые механизмы синхронизации нитей (задач).

Все нити (задачи) разделяют ресурсы и должны обмениваться между собой информацией, поэтому необходимы механизмы межзадачного (межнитиевого) взаимодействия.

К примеру, при доступе к общему ресурсу, такому как GPS-модуль, семафор может использоваться для обеспечения доступа только одной задаче в определенный момент времени, предотвращая конфликты и обеспечивая предсказуемое поведение системы.

Требование 4: Должен существовать механизм наследования приоритетов (система должна быть защищена от инверсии приоритетов).

Под инверсией приоритетов будем понимать изменение их обычного порядка. На самом деле именно эти механизмы синхронизации и тот факт, что разные нити выполняются в одном и том же пространстве памяти, и определяют различие между нитями и процессами. Процессы не разделяют одно и то же пространство памяти.

Комбинации приоритетов нитей и разделение между ними ресурсов приводит к классической проблеме инверсии приоритетов. Для создания условия инверсии приоритетов должно быть задействовано как минимум три

нити. Если нить с самым низким приоритетом заблокировала ресурс (который она делит с самой высокоприоритетной нитью), в то время как работает нить с промежуточным приоритетом, возникает следующий эффект: нить с наивысшим приоритетом ожидает освобождения ресурса; нить с промежуточным приоритетом вытесняет низкоприоритетную нить и работает, пока не завершится; управление получает низкоприоритетная нить, которая освобождает ресурс, и только после этого нить с высоким приоритетом может продолжить свою работу. В этом случае время, необходимое для завершения нити с наивысшим приоритетом, зависит от времени работы нити с более низким приоритетом – это и есть инверсия приоритетов. Очевидно, что в такой ситуации высокоприоритетная нить может "прозевать" критическое событие.

Чтобы избежать таких ситуаций, ОСРВ должна быть снабжена механизмом наследования приоритетов, то есть блокирующая нить должна наследовать приоритет нити, которую она блокирует (конечно, только в том случае, если заблокированная нить имеет более высокий приоритет). Наследование означает, что блокирующий ресурс нить наследует приоритет нити, который она блокирует (это справедливо лишь в том случае, если блокирующая нить имеет более высокий приоритет).

Пример для ИС БСТС: в системе БСТС могут быть реализованы механизмы наследования приоритетов для предотвращения инверсии приоритетов в критических сценариях. Например, предположим, что существуют три нити: нить А с высшим приоритетом, нить В с промежуточным приоритетом и нить С с наименьшим приоритетом.

В данном случае, нить А выполняет задачу мониторинга окружающей среды, нить В отвечает за планирование маршрута, а нить С выполняет задачу управления движением транспортного средства.

Представим ситуацию, где нить С заблокировала общий ресурс, необходимый для нити А, в то время как нить В находится в процессе выполнения. В стандартной ситуации без механизма наследования приоритетов, нить В будет продолжать свое выполнение, а нить А будет

ожидать освобождения ресурса, несмотря на то, что имеет более высокий приоритет. Это может привести к задержкам в реакции на внешние события и потере критических данных.

Однако, благодаря механизму наследования приоритетов, в данном случае нить B наследует приоритет нити A, так как блокирует ресурс, который необходим для выполнения нити A. Таким образом, нить B временно повышает свой приоритет до уровня нити A, позволяя нити A получить доступ к ресурсу независимо от нити C. По окончании использования ресурса, нить B восстанавливает свой исходный приоритет.

Такой подход гарантирует, что высокоприоритетная нить A не будет заблокирована низкоприоритетной нитью C и сможет надлежащим образом реагировать на критические события, такие как обнаружение препятствий на пути или опасных условий.

Требование 5: Поведение ОС должно быть предсказуемо.

Это значит, что во всех сценариях рабочей нагрузки системы должно быть определено максимальное время отклика.

Например, в системе может быть определено, что задача детектирования должна быть выполнена в течение временного интервала $\Delta T = 0{,}3$ с. ОСРВ обеспечивает предсказуемость выполнения задач, что особенно важно для критических по времени задач, таких как обработка данных от датчиков или реакция на препятствия.

Рассмотрим **временные требования** к операционным системам. Разработчик должен знать все времена выполнения системных вызовов и уметь предсказывать поведение системы в любых ситуациях. Поэтому производитель ОСРВ обязательно должен давать информацию о следующих временных характеристиках системы:

- задержке прерывания (interrupt latency) - то есть время от момента появления запроса на прерывание до начала его обработки;
- максимальном времени исполнения каждого системного вызова. Оно должно быть предсказуемым и не зависеть от количества объектов в системе;

- максимальном времени, на которое ОС и драйверы могут блокировать прерывания.

Разработчик также должен знать и учитывать следующее:

- уровни системных прерываний;
- уровни прерываний устройств, максимальное время, которое занимают программы обработки прерываний, и т.д.

Если все перечисленные выше времена известны, то имеются все предпосылки для создания системы жесткого реального времени. При этом требования к производительности разрабатываемой системы должны быть согласованы с характеристиками выбранной ОСРВ и аппаратуры.

Требования, накладываемые на вычислительную установку реального времени, формулируются следующим образом:

1. В зависимости от сложности программы управления, требование «реального времени» накладывает различные условия на вычислительную мощность процессора для СРВ.

2. Внешние события становятся известны системе посредством прерываний (interrupt requests (IRQ)) (т.е. запросов на обслуживание со стороны внешних устройств). Поэтому часто для ОСРВ более важна не мощность процессора, а характеристики компьютера, связанные с подсистемой прерываний. Желательными являются:

- наличие как можно большего количества уровней прерываний (IRQ levels) (т.е. аппаратного или/и программного декодирования источника запроса);

- как можно меньшее время реакции на прерывание (т.е. как можно меньшее время между поступлением запроса на обслуживание и началом выполнения обслуживающей программы).

3. СРВ часто сама является инициатором периодических процессов, которыми управляет. Поэтому необходимо иметь в наличии один или несколько таймеров (аппаратных устройств, выдающих прерывание через

заданные промежутки времени), которые могут работать в периодическом или ждущем режиме.

4. Ввиду того, что СРВ часто управляет ответственными промышленными процессами, данное обстоятельство выдвигает очень жесткие требования к надежности используемого оборудования.

В результате исследования и разработки вычислительной подсистемы БСТС была разработана архитектура, позволяющая снять ограничения, связанные с ограниченностью ресурсов и строгой приоритезацией задач (Рисунок 2.5):

Рисунок 2.5 – Вычислительные блоки 1) низкого, 2) среднего и высокого уровня

Разработанная вычислительная подсистема БСТС позволяет эффективно управлять задачами, снять ограничения, связанные с ограниченностью ресурсов и строгой приоритезацией. Она обеспечивает гибкость, адаптивность и оптимальное использование вычислительных ресурсов, что способствует повышению эффективности работы БСТС и обеспечивает надежность и точность выполнения задач.

2.2.2 Снятие системного ограничения в подсистеме коммуникации и связи ИС БСТС

В разработке подсистемы коммуникации и связи ИС БСТС требуется уделить особое внимание нескольким фундаментальным аспектам, которые оказывают прямое воздействие на всю систему. Эти ключевые аспекты включают в себя пропускную способность каналов связи, задержки и латентность в передаче данных, помехоустойчивость, безопасность данных и совместимость с различными устройствами. Ниже мы предложим методы и механизмы для преодоления данных ограничений, что в конечном итоге обеспечит более эффективную и надежную работу ИС БСТС:

Беспроводные технологии нового поколения (5G)

Технология 5G внесла значительные усовершенствования в области беспроводных коммуникаций по сравнению с предыдущими поколениями (2G, 3G, 4G), и это имеет непосредственное отношение к снятию системных ограничений в подсистеме коммуникации и связи ИС БСТС.

При описании преимуществ использования этой технологии мы будем говорить не о всей технологии как таковой, а о конкретной реализации и решении - набор 5G Development Kit от «МТС» (Рисунок 2.6).

Рисунок 2.6 – Комплект 5G Development Kit

Основные характеристики комплекта 5G Development Kit[41]:

- совместимость с 5G-модулями мировых производителей: Quectel, Telit, Fibocom, SimCom и т.д;
- использование антенн 5G sub-6GHz, GNSS и 5G mmWave диапазона, возможно подключение антенн собственного производства и их размещение на корпусе;
- поддержка спецификаций 3GPP 4G/5G Rel. 15;
- динамическое распределение спектра между 5G и 4G;
- максимальная выходная мощность передатчика LTE/5G: 26 dBm (Power Class 2);
- подключение к сети с помощью SIM-карты или eSIM-чипа (в формате MFF2);
- поддержка голосовых режимов передачи: VoLTE, VoNR;
- аппаратная совместимость с одноплатным компьютером Raspberry Pi;
- поддержка работы по USB-порту с ОС Windows 10, Linux, Android;

• подключение к ПК и другим устройствам через кабель USB type C — type A (USB 3.1 Gen 2);

• дополнительные интерфейсы: PCIe Gen 3×1-lane, I2C, GPIO, Digital audio (I2S и PCM);

• возможность подключения кулера и размещения пассивного радиатора;

• работа от сетевого блока питания или аккумуляторов.

С точки зрения снятия системных ограничений в подсистеме коммуникации и связи ИС БСТС данный комплект обладает следующими характеристиками и преимуществами:

1. Высокая пропускная способность: 5G обеспечивает значительно более высокую пропускную способность по сравнению с предшествующими стандартами. Это означает, что большее количество данных может быть передано через сеть за более короткий промежуток времени. Для БСТС это означает, что она способна обрабатывать и передавать большой объем информации, включая видео, сенсорные данные и управляющие команды, без значительных задержек.

2. Низкая латентность: 5G минимизирует задержки в передаче данных. Это важно для реального времени в БСТС, где даже малейшие задержки могут повлиять на безопасность и эффективность системы. Низкая латентность позволяет БСТС мгновенно реагировать на изменяющиеся условия окружающей среды.

3. Большая емкость сети: 5G расширяет возможности сети для подключения большого количества устройств. Это важно для БСТС, где большое количество сенсоров и устройств должны взаимодействовать между собой и с центральным управлением.

4. Помехоустойчивость и надежность: 5G включает механизмы самоисправления и переключения на более надежные частоты и каналы при возникновении помех. Это обеспечивает надежную связь и помогает предотвратить потерю данных в условиях воздействия внешних факторов.

5. Поддержка высокой плотности устройств: 5G может поддерживать большое количество подключенных устройств на единицу площади. Для БСТС это важно, так как в сельскохозяйственных операциях может быть много сенсоров и управляющих устройств на определенной территории.

6. Миниатюризация и низкое энергопотребление: 5G устройства и модули становятся всё более компактными и энергоэффективными, что подходит для интеграции в БСТС.

Таким образом, технология 5G может быть использована в подсистеме коммуникации и связи ИС БСТС для обеспечения высокой пропускной способности, низкой латентности, помехоустойчивости и надежности, а также поддержки большого количества устройств. Все эти характеристики тесно связаны с решением системных ограничений, таких как пропускная способность каналов связи, латентность и задержки, помехоустойчивость, и защита данных, что делает 5G ключевой технологией для проектирования ИС БСТС.

Использование протокола SCTP

Для обеспечения высокоскоростной связи 5G, требуется применение надежного и высокоэффективного протокола транспортного уровня.

На данный момент существует три основных протокола и семейств протоколов транспортного уровня: UDP, TCP и SCTP. Из них выделяют два семейства UDP и TCP протоколов. Далее рассмотрим эти протоколы передачи данных и проведем их анализ.

UDP семейство:

В данном семействе протоколов выделим UDP(RFC-768), UDP-Lite, R-UDP:

1. UDP(RFC-768). Протокол был создан в 1980 году Дэвидом П. Ридом и официально определён в RFC-768 (спецификацию). UDP использует простую модель передачи, без явных «рукопожатий» для обеспечения надёжности, упорядочивания или целостности данных. Датаграммы могут

прийти не по порядку, дублироваться или вовсе исчезнуть без следа, но гарантируется, что если они придут, то в целостном состоянии. UDP подразумевает, что проверка ошибок и исправление либо не нужны, либо должны исполняться в приложении.

Природа UDP как протокола без сохранения состояния также полезна для серверов, отвечающих на небольшие запросы от огромного числа клиентов, например DNS и потоковые мультимедийные приложения вроде IPTV, Voice over IP, протоколы туннелирования IP и многие онлайн-игры.

2. UDP-Lite. Беспроводные каналы обычно имеют более низкую скорость передачи данных и более высокую частоту ошибок по сравнению с проводными каналами. Такие приложения, как потоковая передача аудио и видео, могут использовать устойчивые к ошибкам кодеки, но чувствительны к задержкам. Следовательно, повторная передача из-за ошибок контрольных сумм является дорогостоящей. Для этих классов приложений был разработан UDP Lite. Он используется для потоковой передачи мультимедиа, а также видео через мобильные телефоны, с хорошими результатами.

В отличие от UDP, в котором пакет или целиком защищён контрольной суммой (checksum) или весь пакет не защищён, UDP Lite допускает возможность частичных контрольных сумм, которые покрывают только часть датаграммы, и таким образом возможна доставка частично поврежденных пакетов. Это было создано для мультимедийных протоколов, у которых прием пакета с частично поврежденной полезной нагрузкой считается более предпочтительным вариантом, нежели не получение пакета вовсе.

3. R-UDP. Протокол был разработан в Bell Labs для операционной системы Plan 9. R-UDP направлен на предоставление решения, в котором UDP слишком примитивен, так как доставка пакетов гарантированного порядка является желательной, но TCP добавляет слишком много сложности/накладных расходов. Чтобы улучшить качество обслуживания, UDP реализует функции, аналогичные TCP, но с меньшими накладными расходами.

Для обеспечения качества он расширяет UDP, добавляя следующие функции:

- Подтверждение полученных пакетов
- Управление окнами и потоками
- Повторная передача потерянных пакетов
- Чрезмерная буферизация (быстрее, чем потоковая передача в реальном времени)

В общем семейство протоколов UDP отличается тем, что использует связь без установки соединения. Из-за чего отправитель не знает готов ли клиент к приему данных и получил ли он пакет. В итоге данный вид связи позволяет передавать информацию довольно быстро, но из-за отсутствия подтверждения можно потерять часть информации.

Семейство протоколов TCP:

В отличие от семейства UDP семейство протоколов TCP требует установления соединения перед началом передачи данных, т.е. клиент и сервер должны подключиться друг к другу и договориться о том каким образом будет осуществляться передача. TCP поддерживает отправку и прием подтверждений, обработку тайм-аутов, повторную передачу, управление потоком и другие возможности.

В данном семействе протоколов выделим такие виды, как TCP (RFC768), TCP-Lite, Multipath TCP:

1. TCP (RFC768). Протокол управления передачей TCP (Transmission Control Protocol), разработанный по заказу агентства перспективных исследовательских программ ARPA, был опубликован в документе RFC 793 в сентябре 1981 года, а уже в 1983 полностью заменил собой протокол NCP (Network Control Protocol) в сети ARPANET (прародителей Интернета) и сегодня стал основным протоколом для передачи данных.

Протокол TCP перед началом передачи данных в обязательном порядке устанавливает соединение и обеспечивает использующим его приложениям надежный упорядоченный двухсторонний байтовый поток.

Основные интернет-приложения, такие как Всемирная паутина, электронная почта, удаленное администрирование и передача файлов, полагаются на TCP.

2. TCP-Lite. TCP-Lite предоставляет упрощенную версию TCP, которая подавляет передачу некоторых TCP блоки данных протокола, которые используются для согласования создания и закрытия сеансов. В то же время TCP-Lite передает все данные приложения из исходного канала, сохраняя порядок, целостность, надежность и безопасность исходного TCP-транспорта.

3. Multipath TCP. Был создан в январе 2013 года IETF и опубликован в спецификации Multipath в качестве экспериментального стандарта в RFC 6824. В марте 2020 года он был заменен спецификацией Multipath TCP v1 в RFC 8684. Данный протокол был создан с целью получения многопоточности в TCP подобных протоколах.

Многоканальный TCP был разработан для обратной совместимости с обычным TCP. Таким образом, он может поддерживать любое приложение. Тем не менее, некоторые специфические развертывания используют возможность одновременного использования разных путей.

SCTP:

SCTP протокол создает двустороннее сопоставление между двумя конечными точками и позволяет работать с несколькими потоками для каждой пары конечных точек, а также обеспечивает поддержку концепции многоинтерфейсного узла в транспортный уровень.

Как и TCP семействе протоколов, протокол SCTP обеспечивает надежность, последовательность, управление потоком и полнодуплексную передачу данных.

В отличие от TCP семейства, протокол SCTP обеспечивает:

- Ассоциация вместо "соединения": ассоциация относится к связи между двумя системами, которая может включать более двух адресов из-за множественной адресации.

• Ориентированный на сообщения: обеспечивает последовательную доставку отдельных записей. Как и в случае с UDP семейством, длина записи, записанной отправителем, передается принимающему приложению.

• Множественная адресация: позволяет одной конечной точке SCTP поддерживать несколько IP-адресов. Эта функция может обеспечить повышенную устойчивость к сбоям в сети.

Сравнение семейств протоколов UDP и TCP с SCTP:

SCTP объединяет в себе положительные черты UDP и TCP. Основным преимуществом SCTP протокола является:

• Сохранение границ – в семействе протокола TCP сообщения передаются непрерывным потоком байтов, читаются аналогично, поэтому программист должен сам расставлять вручную в полученном пакете границы и получать оттуда куски данных. SCTP позволяет ставить границы и обрабатывать данные целыми пакетами, как и в UDP семействе. Но при этом гарантирует порядок доставки пакетов, обеспечивает надёжность и ориентирован на соединения.

• Поддержка множественных интерфейсов - SCTP позволяет установить связь с одним сервером через несколько разных линий связи, к примеру по Wi-Fi и Ethernet. Таким образом если одна из линий связи обрушиться, то соединение не разорвется. Так же эта функция позволяет увеличить скорость передачи данных. Если же в TCP оборвется линия связи, то связь будет потеряна.

• Многопоточность – позволяет в рамках одной ассоциации передавать несколько потоков данных. В TCP данные и служебная информация передаются по одному соединению, из-за чего могут возникать задержки в получение информацией в связи с текущей передачей данных.

• В SCTP нет полузакрытых соединений, как в TCP. Если мы закроем связь, то сразу в обоих направлениях.

Сравнение функциональных возможностей рассмотренных протоколов представлено в Таблице 2.1:

Таблица 2.1 – сравнение функциональных возможностей семейств протоколов UDP, TCP и SCTP.

Функция	UDP	UDP-Lite	R-UDP	TCP	TCP-Lite	Multipath TCP	SCTP
Установка соединения	Нет	Нет	Нет	Да	Да	Да	Да
Поддержка дуплексного режима	Да	Да	Да	Да	Да	Да	Да
Надежная передача	Нет	Нет	Нет	Да	Да	Да	Да
Частично надежная передача	Нет	Да	Да	Нет	Нет	Нет	Опционально
Упорядоченная доставка	Нет	Нет	Нет	Да	Да	Да	Да
Неупорядоченная доставка	Да	Да	Да	Нет	Нет	Нет	Да
Контроль потока	Нет	Нет	Да	Да	Да	Да	Да
Управление потоком	Нет	Нет	Нет	Да	Да	Да	Да
Уведомление о перегрузке	Нет	Нет	Да	Да	Да	Да	Да
Выборочное подтверждение	Нет	Нет	Нет	Опционально	Опционально	Опционально	Да
Сохранение границ сообщения	Да	Да	Да	Нет	Нет	Нет	Да
Многопоточность	Нет	Нет	Нет	Нет	Нет	Да	Да
Защита от SYN-flood-атак	-	-	-	Нет	Нет	Нет	Да
Поддержка множественных интерфейсов	Нет	Нет	Нет	Нет	Нет	Да	Да
Контроль работоспособности	Нет	Нет	Нет	Нет	Нет	Нет	Да
Скорость передачи	Высокая	Высокая	Высокая	Низкая	Высокая	Низкая	Средняя

Исходя из выше приведённого анализа больше всех преимуществ имеет SCTP протокол связи. Он соединил в себе лучшие черты TCP и UDP протоколов, благодаря чему смог осуществить не только надежную связь, но и сохранять границы сообщения как в UDP. Так же для ускорения передачи сообщения SCTP использует многопоточность и поддержку множественных интерфейсов, но, чтобы не происходили сбои в передачи информации SCTP использует контроль работоспособности протокола, а также защиту от SYN-flood-атак для обеспечения безопасной передачи данных.

Таким образом, протокол SCTP предоставляет набор функций и характеристик, которые снимают несколько системных ограничений в подсистеме коммуникации и связи ИС БСТС, такие как пропускная способность, надежность, устойчивость к помехам, защиту данных и совместимость с разными устройствами. Этот протокол дополняет и усиливает возможности ИС БСТС, делая ее более эффективной и надежной в реальных сельскохозяйственных условиях.

2.2.3 Снятие системного ограничения в подсистеме позиционирования и навигации ИС БСТС

Для обеспечения точного позиционирования и навигации БСТС в различных сельскохозяйственных условиях, использование современных аппаратно-программных средств становится необходимостью. Однако при разработке подсистемы позиционирования и навигации существуют системные ограничения, которые необходимо учитывать. В данном разделе мы рассмотрим технологии, способные снять эти ограничения, сделав систему более точной и надежной.

Для формирования навигационно-временного поля БСТС предлагается использовать высокоточные аппаратно-программные средства, которые опираются на сигналы глобальных навигационных спутниковых систем (ГНСС). Среди технологий, соответствующих требованиям к точности позиционирования (указанных в главе 1), выделяются RTK (Real Time Kinematic) и PPP (Precise Point Positioning). RTK, несомненно, предоставляет высокую точность, однако ограничивает область использования действием базовых станций ГНСС. Это ограничение можно считать одним из системных ограничений, так как оно связано с радиусом действия базовых станций.

Технология PPP является одной из наиболее передовых технологий абсолютной высокоточной навигации подвижных объектов на открытых пространствах и рассматривается как наиболее перспективная. В основе данной технологии лежит использование информации о точных орбитах спутников ГНСС и поправок ко времени излучения спутниковых сигналов.

Погрешности бортовых эфемерид ГНСС, используемых в обычных приемниках ГНСС, имеют порядок в несколько метров, и не могут быть устранены (в отличие от всех остальных типов погрешностей при приеме спутниковых сигналов) в одном приемнике без привлечения дополнительной информации (например, дифференциальных методов навигации).

Данный вид поправок (PPP) получается путем обработки измерительной информации от сетей референсных станций ГНСС в специализированных центрах, и передается пользователям во времени, близком к реальному. За рубежом известен ряд сервисов, обеспечивающих передачу точных эфемерид через геостационарные спутники. В Российской Федерации в перспективе точная эфемеридно-временная информация будет передаваться непосредственно со спутников ГЛОНАСС по основным каналам передачи (в диапазоне частот L3). В настоящее время, в общем случае, поправки для реализации навигации в режиме PPP могут быть получены через сеть Интернет (с серверов соответствующих центров обработки), либо локальную беспроводную сеть объекта служебного (телематического) или общего пользования.

Для получения высоких точностей местоопределения объектов (дециметрового и субдециметрового уровня) с использованием поправок PPP в потребительском приемнике ГНСС реализуется обработка измерительной информации не только по коду, но и по фазе несущей. Существенным преимуществом данного подхода является возможность обеспечения точной навигации объектов при помощи только одного пользовательского приемника ГНСС на любых территориях, не оборудованных инфраструктурой дифференциального сервиса (базовыми станциями).

Использование технологии PPP позволяет снять ограничения, связанные с точностью позиционирования. Эта технология обеспечивает высокую точность на больших территориях, что особенно важно в сельском хозяйстве, где разнообразные задачи могут потребовать точного позиционирования сельскохозяйственной техники.

К сожалению, решение задачи определения точного навигационно-временного поля в автономном режиме с использованием только технологии PPP имеет следующие неустранимые недостатки:

1. Невозможность определения точных навигационных оценок в случае недоступности и-или высокой латентности приема сообщений с уточненной эфемеридно-временной информации PPP.

2. Нестабильность получения точных навигационных оценок в связи с неоднозначностью псевдодальностных оценок с видимых спутников в условиях многолучевого распространения (в т.н. «городских каньонах» - плотной заводской застройке, при проезде у препятствий с металлической окантовкой и пр.).

3. Невозможность функционирования в условиях отсутствия видимости открытого неба в большей части верхней полусферы (например, при частичном затенении сигналов ГНСС высокими деревьями, в закрытых помещениях и пр.).

4. Невозможность функционирования в условиях индустриальных и искусственно индуцированных помех на частотах L1 и L2 – диапазона.

5. Малая частота обновления навигационных оценок – до 10 Гц – связанная с высокой вычислительной сложностью фазовой обработки одновременно с появлением неоднозначностей в решении навигационно-временной задачи на меньших временных интервалах.

6. Невозможность получения однозначной ориентации ТС.

7. Отсутствие точной первичной информации о динамике движения ТС (угловых скоростях, линейных скоростях и ускорениях).

Совмещение абсолютной системы навигации ГНСС в режиме PPP с относительной (инерциальной) системой навигации позволяет устранить все вышеупомянутые недостатки, обеспечивая:

1. Определение точных относительных (относительно последней релевантной точки, полученной абсолютной системой навигации)

навигационных оценок в случае полной недоступности каналов связи и-или всего радиоэфира (в пределах заявленного периода автономности).

2. Определение точных относительных навигационных оценок в условиях полной или частичной недоступности спутниковой навигационной группировки (в закрытых помещениях, при неблагоприятных условиях наблюдения группировки ГНСС – в пределах заявленного периода автономности).

3. Полное непрерывное комплексирование с контуром ГНСС – с получением непрерывно уточняемого поля навигационных оценок и точной первичной информации о ориентации и динамике движения БСТС (угловых скоростях, линейных скоростях и ускорениях).

Далее предлагается решение («Созвездие»), разработанное с участием автора[42]. На Рисунке 2.7 представлена общая структурная схема предлагаемого решения. Система имеет в своем составе инерциальную подсистему (2 инерциальных сборки), ГНСС подсистемы (ГНСС приемник и ГНСС антенна) и вычислительного блока.

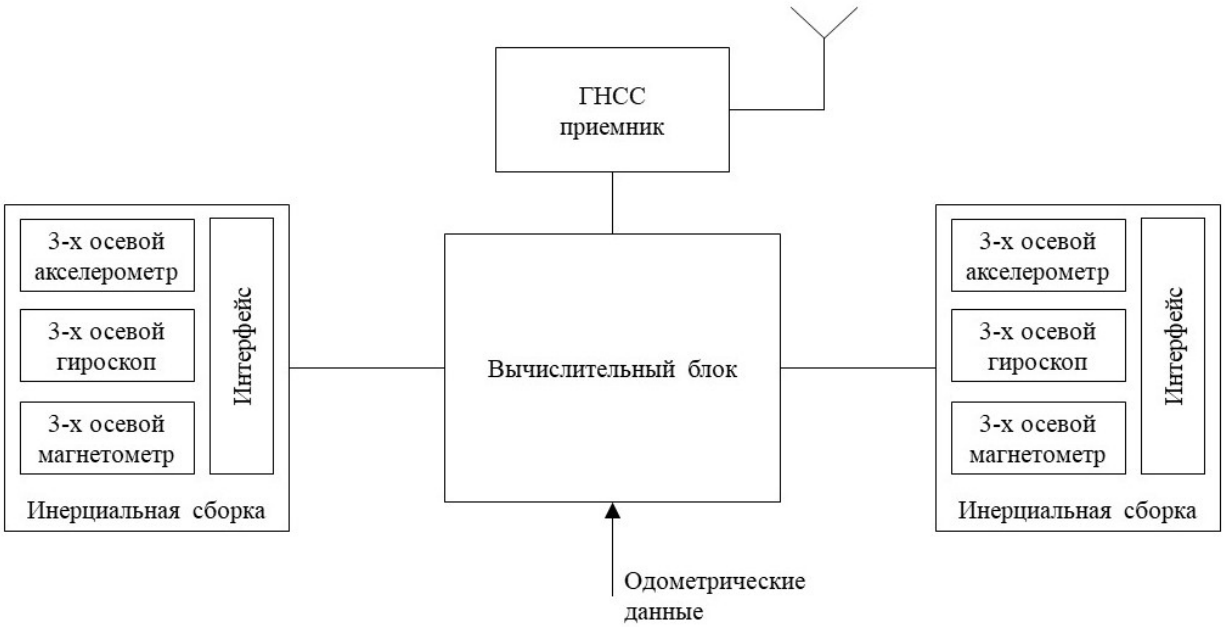


Рисунок 2.7 – Структурная схема совмещенной навигационной системы

Инерциальная подсистема позволяет получать навигационные оценки (координаты и углы поворота транспортного средства) при плохом качестве

или полном отсутствии ГНСС сигнала. Каждая инерциальная сборка имеет в своем составе 3-х осевой гироскоп, 3-х осевой акселерометр, 3-х осевой магнитометр и трансивер интерфейса передачи данных. Инерциальные сборки осуществляют измерение линейных ускорений и угловых скоростей с частотой до 100 Гц и передают данные на вычислительный блок.

ГНСС подсистема построена на базе двухчастотного ГНСС приемника, работающего в режиме PPP с частой до 10 Гц с точностью определения координат не менее 10 см.

Вычислительный блок принимает данные со всех внешних устройств, осуществляет фильтрацию и вычисление координат ИНС, декодирование данных ГНСС приемника, а также своевременное переключение с ГНСС навигации на инерциальную и наоборот.

Одновременно с вычислением навигационных оценок по данным инерциальной подсистемы, вычислительный блок осуществляет прием и анализ данных с ГНСС приемника. При «хорошем» уровне ГНСС сигнала (СКО координаты в горизонтальной плоскости не превышает 0,1 м), выходными координатами совмещенной навигационной системы являются спутниковые координаты. В этом режиме происходит постоянная коррекция рассчитанных координат инерциальной подсистемы, а также корректировка курса по данным ГНСС при движении транспортного средства. Таким образом, реализовано отсутствие накопления ошибки подсистемой инерциальной навигации при наличии достаточного уровня ГНСС сигнала. При ухудшении качества спутникового сигнала (при уменьшении СКО координаты менее 0,1 м) происходит бесшовное переключение выхода системы на инерциальную навигацию.

Спутниковый приемник позволяет работать в следующих режимах: одиночный, RTK и PPP с получением поправок спутниковых орбит через сеть Интернет. Обмен данными с вычислительным блоком реализован по протоколу BINR по интерфейсу Ethernet. ГНСС приемник имеет быстрое

время старта (менее 1 минуты) и обеспечивает выдачу точных координат с темпом до 2 Гц.

В основе блока инерциальной навигации датчик абсолютной ориентации BMX160 с частотой выборки 100 Гц.

Вычислительный блок реализован на основе микроконтроллера семейства STM32H745 с реализацией необходимых интерфейсов для подключения внешних устройств. Выбор данного микроконтроллера обусловлен его большой распространенностью, хорошей документированностью, а также достаточным быстродействием для выполнения необходимых математических операций в реальном времени.

На Рисунке 2.8 приведена фотография системы навигации «Созвездие»:

Рисунок 2.8 – Система высокоточной спутниково-инерциальной навигации «Созвездие»

Система "Созвездие" успешно снимает следующие системные ограничения, описанные в контексте подсистемы позиционирования и навигации в ИС БСТС:

1. **Точность позиционирования**: одним из основных ограничений была необходимость обеспечения высокой точности позиционирования БСТС.

Система "Созвездие" достигает высокой точности позиционирования за счет технологии PPP (Precise Point Positioning) при использовании двухчастотного ГНСС приемника. Это обеспечивает точность координат не менее 10 см, преодолевая данное ограничение.

2. **Доступность сигналов навигационных спутников**: ограничение, связанное с доступностью сигналов ГНСС, было решено системой "Созвездие" путем интеграции инерциальных сборок. Даже в случаях, когда сигналы ГНСС недоступны или имеется ограниченная видимость спутников, система продолжает надежно работать. Это достигается благодаря способности инерциальных сборок измерять линейные ускорения и угловые скорости транспортного средства без зависимости от доступности сигналов навигационных спутников. Таким образом, система "Созвездие" успешно снимает ограничение, связанное с доступностью сигналов ГНСС.

3. **Зависимость от глобальных систем навигации**: ограничение, связанное с зависимостью от глобальных систем навигации, было успешно преодолено так же путем интеграции инерциальной подсистемы. При наличии ГНСС сигналов система активно использует спутниковые координаты для формирования навигационных оценок. Тем не менее, в случае недоступности сигналов ГНСС или сбоев в их работе, система легко переключается на инерциальную навигацию. Это снимает зависимость от работы глобальных систем навигации и обеспечивает надежность навигации даже при их отсутствии.

Таким образом, система "Созвездие", разработанная с непосредственным участием автора, представляет инновационное решение, которое успешно преодолевает ограничения, связанные с точностью позиционирования, доступностью сигналов ГНСС и зависимостью от глобальных систем навигации. Ее совмещенный подход, интегрирующий инерциальную и ГНСС-навигацию с использованием PPP, обеспечивает надежную, точную и непрерывную навигацию в самых разнообразных условиях функционирования.

2.2.4 Снятие системного ограничения в подсистеме машинного зрения ИС БСТС

При проектировании подсистемы машинного зрения в ИС БСТС для сельского хозяйства, среди основных задач стоит задача эффективного обнаружения и классификации разнообразных препятствий, таких как столбы, деревья, кустарники, овраги и другие объекты. Эти задачи играют ключевую роль в обеспечении успешных сельскохозяйственных операций, таких как навигация машин и оборудования, анализ почвы и состояния растений, а также обеспечение безопасности при работе с сельскохозяйственными машинами.

Для успешного снятия системных ограничений (указанных выше в предыдущем подразделе) и обеспечения высокой эффективности работы подсистемы машинного зрения в сельском хозяйстве, далее будут описаны методы детектирования и классификации, а так же представлены алгоритмы нормализации изображений. Эти алгоритмы включают в себя оценку уровня шума, коррекцию резкости, компенсацию засветки и контрастности, а также алгоритмы детектирования и классификации объектов, и рассмотрены в [28]. Такой подход обеспечит высокую точность, надежность и эффективность в работе подсистемы машинного зрения, позволяя эффективно справляться с задачами обнаружения и классификации препятствий в условиях сельского хозяйства.

Алгоритм нормализации изображений

Применение необработанных видеопотоков и изображений в алгоритмах часто ограничивает достижение необходимой точности в процессе обнаружения и классификации из-за низкого качества визуальных характеристик изображения, таких как низкая резкость, контрастность и наличие шумов. Это также увеличивает время обработки из-за наличия избыточных данных, например, цветовых каналов или высокого разрешения, что делает невозможным использование таких методов в реальном времени.

Особенностью подсистемы машинного зрения в ИС БСТС является использование алгоритма автоматической оценки и предварительной

обработки изображений. Этот метод позволяет значительно улучшить качество обнаружения и классификации объектов, особенно в условиях ограниченной видимости и переменной окружающей среды. Далее рассмотрим критерии, процессы вычислений и методики сбора данных, необходимые для разработки этих критериев с учетом ранее определенных системных ограничений.

Алгоритмы нормализации изображений разработаны и подробно рассматриваются в исследовании [28] и будут использованы в данной работе в качестве шаблона проектирования. Далее кратко рассмотрим основные результаты.

Уровень шума

Оценка шума является критическим этапом в различных алгоритмах обработки и анализа изображений. Точное количественное измерение шумовых компонент позволяет настраивать предобработку в зависимости от фактического уровня шума, вместо использования статических пороговых значений. В данном алгоритме применяется метод оценки шума, подробно описанный в [43], но также существуют альтернативные варианты его реализации.

Уровень резкости

Резкость изображения характеризует, насколько детали четко различимы на фотографии и как четко выделены грани между участками изображения с разной яркостью. Уровень резкости напрямую зависит от изменений яркости в пространстве. В самом общем понимании, резкость изображения S в определенной точке можно интерпретировать как градиент яркости I, что означает, что резкость изображения представляет собой векторное поле ($S = \nabla I$).

В процессе анализа входного изображения используется функция для оценки уровня резкости, которая измеряет резкость и обозначается как SL. Данный метод [44] основан на вычислении модуля градиента изображения. Значения SL могут быть интерпретированы следующим образом: значения

менее 2 указывают на низкую резкость изображения, значения от 2 до 4.5 - на среднюю резкость, а значения более 4.5 - на высокую резкость.

Изображения с высокой степенью резкости не требуют предварительной обработки. Для значений SL в диапазоне от 0 до 4.5 используется фильтр размытия с индивидуальными параметрами, которые определяются для каждого типа оптического сенсора.

Наличие засветки

Засветка на изображении определяется наличием переэкспонированных областей, резких теней и областей с низким контрастом. Это может значительно повлиять на последующее распознавание объектов на изображении. Для определения наличия засветки, обозначаемой как FL, используется специальная функция.

Если на изображении обнаружена засветка ($FL = 1$), то применяется фильтр адаптивного гистограммного выравнивания.

Уровень контрастности

Контрастность измеряет разницу в яркости или цвете, которая позволяет выделить объекты на изображении. Уровень контрастности на изображении измеряется и обозначается как CL. Значения CL в диапазоне от 0 до 70 считаются низкими контрастностями, в то время как $CL > 70$ указывает на высокую контрастность.

Для изображений с высокой контрастностью ($CL \geq 70$) не применяется фильтрация по контрастности. В случае, если CL находится в диапазоне от 0 до 70, применяется фильтр CLAHE (алгоритм контрастно-ограниченной эквализации гистограммы изображения).

Алгоритм нормализации

На Рисунке 2.9 представлена блок-схема алгоритма нормализации изображений [28]. Чтобы избежать избыточных вычислительных затрат и оптимизировать время выполнения алгоритма, для каждой функции оценки была установлена частота применения. Эта частота определяется на основе изменений в свойствах изображения в видеопотоке и потребностей

алгоритмов последующего распознавания. В Таблице 2.2 указаны порядок и частота применения функций оценки критериев [45, 46].

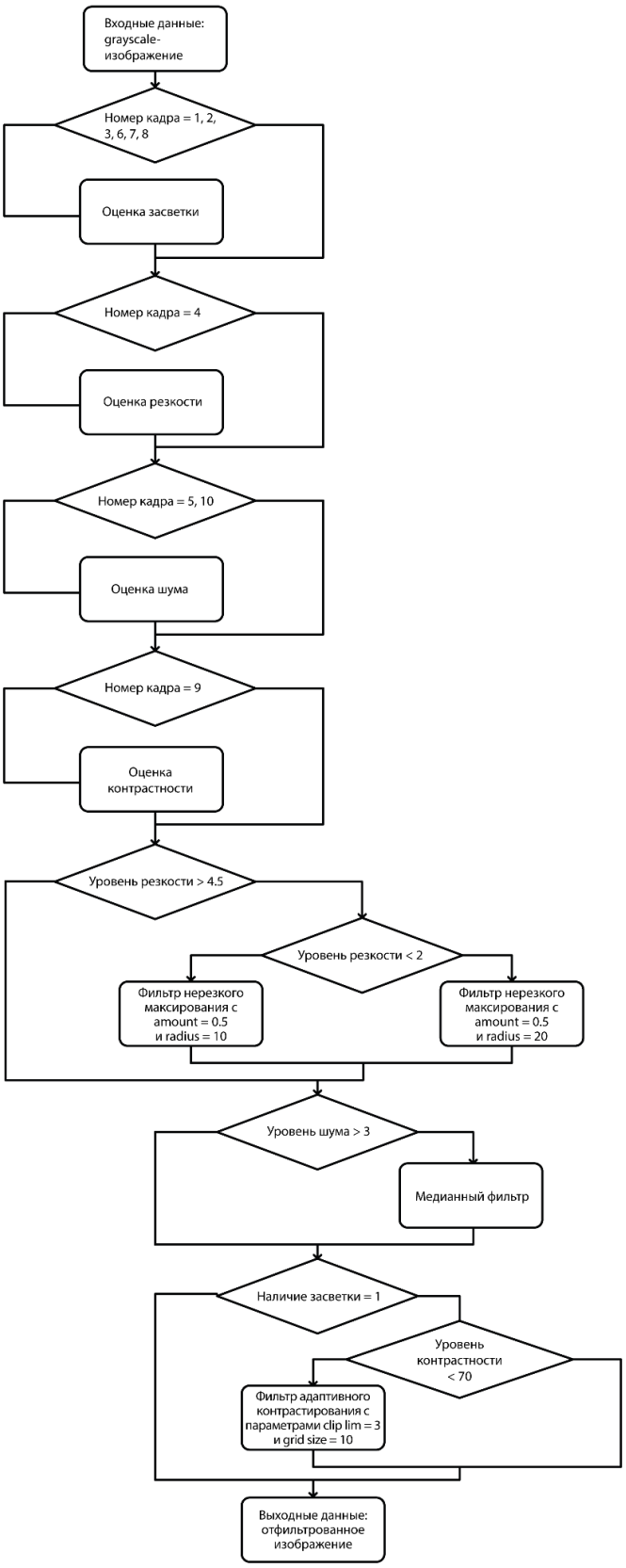


Рисунок 2.9 – Алгоритм нормализации изображений

Таблица 2.2 – Частота применения функций.

Порядок применения	Функция	Частота применения (раз в секунду)
1	Оценка уровня шума	6
2	Оценка уровня резкости	3
3	Наличие засветки	18
4	Оценка уровня контрастности	3

Модуль обнаружения и распознавания объектов

В условиях работы на поле, основными препятствиями, с которыми сталкивается сельскохозяйственная техника, являются столбы, деревья и кустарниковые насаждения. Эти препятствия могут создавать ряд проблем и вызывать опасность при выполнении различных операций в сельском хозяйстве. Столбы, такие как электрические столбы или заборы, могут ограничивать свободу перемещения сельскохозяйственной техники. При неправильном управлении или отсутствии достаточной видимости, столбы могут быть повреждены или даже повалены, что может привести к серьезным последствиям для оборудования, оператора и окружающей среды.

Деревья и кустарниковые насаждения также представляют определенные проблемы на поле. Они могут создавать узкие проходы или перекрыть доступ к определенным участкам. Для сельскохозяйственной техники может быть сложно маневрировать вокруг деревьев и кустарников, особенно если они расположены плотно или имеют непредсказуемую форму.

Преодоление данных препятствий требует точного обнаружения и оценки их местоположения и размеров [47]. Далее предлагается техническое решение, основанное на системе машинного зрения, которое может быть применено для обнаружения и классификации данных препятствий.

Учитывая технологические параметры использования сельскохозяйственных машин с участием автора был разработан аппаратно-программный комплекс сегментации объектов препятствий[48]. По итогам исследования [49-65] темы детектирования [66] и классификации [67]

объектов на изображениях в рамках данной работы используется сверточная нейронная сеть U-Net.

Нейронная сеть U-Net представляет собой эффективную архитектуру глубокого обучения, специально разработанную для задач сегментации изображений. Ее особенностью является наличие пути прямого связывания между низкоуровневыми и высокоуровневыми признаками, что позволяет сети успешно извлекать и сохранять мелкие детали и контекст информации при работе с изображениями.

Применение нейронной сети U-Net позволяет достичь высокой точности и эффективности в обнаружении и классификации различных типов препятствий на поле, таких как столбы, деревья и кустарниковая растительность. Благодаря возможности работы в реальном времени, алгоритмы, основанные на U-Net, способны оперативно обрабатывать видеопотоки с камер и предоставлять точную информацию о препятствиях.

Для обработки изображений препятствий на поле, таких как столбы, деревья и кустарниковая растительность, была использована нейронная сеть с настроенной архитектурой U-Net, изображенной на Рисунке 2.10. Входные данные подавались в виде тензора размерностью 512×512×3, представляющего собой изображение с трех каналов RGB.

Такая архитектура U-Net позволяет эффективно извлекать и сохранять информацию о препятствиях на различных уровнях детализации. Это особенно полезно при распознавании и классификации различных типов препятствий на поле. Последовательность операций свертки, нормализации и активации помогает выявлять ключевые признаки и структуры препятствий, обеспечивая точность и надежность в процессе обработки изображений.

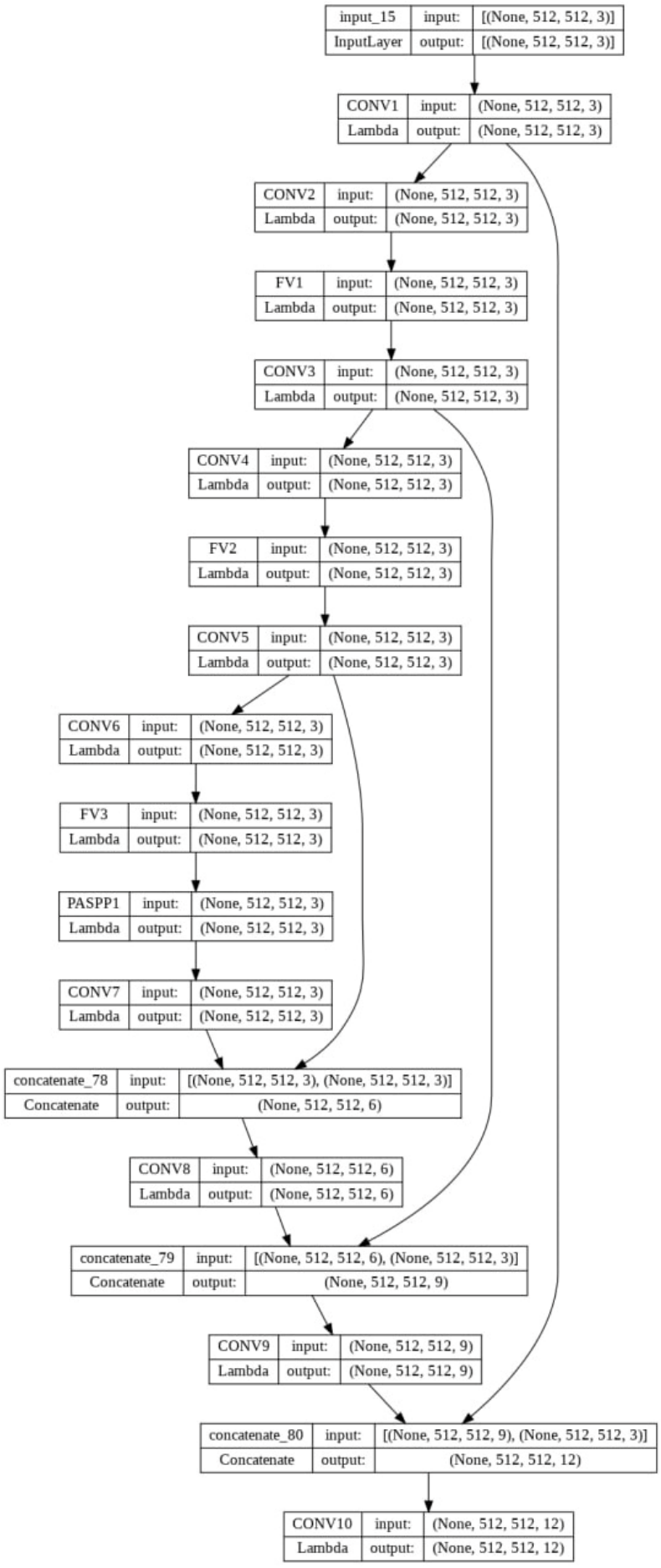


Рисунок 2.10 – Архитектура U-Net

В целях реализации разработанного алгоритма был разработан и изготовлен аппаратно-программный комплекс [68], включающий цветную матричную камеру промышленного сканирования JAI GO-5000C-PGE и вычислитель с характеристиками: ОЗУ 16 ГБ, видеопамять 6 ГБ NVIDIA GeForce GTX 1660 Ti MAX-Q.

Для реализации системы машинного зрения автором была использована технология ускорения вычислений на базе векторных ускорителей CUDA [69].

Так же была разработана программа на языке Python с использованием нейронной сети. Обучение проводили на выборке из 10000 изображений, предварительно размеченных в целях сегментации препятствий типа столб, дерево и древесно-кустарниковая растительность. В рамках испытаний разработанного аппаратно-программного комплекса были проведены 10 групп экспериментов длительностью по 10 минут каждый по детектированию и классификации объектов классов «столб», «дерево», «кустарниковая растительность». При проведении испытаний были определены интервал времени обнаружения $\Delta T = 0{,}5$ с и интервал дальности обнаружения до $\Delta S = 40$ м. Результаты проведения испытаний распознавания (детектирование и классификация) представлены в Таблицах 2.3 и 2.4. За нулевую гипотезу принято отсутствие целевого объекта на кадре: ошибка I-го рода соответствует ложной тревоге, ошибка II-го рода – пропуску цели.

Таблица 2.3 – Результаты детектирования объектов.

Вид ошибки	Номер испытания									
	1	2	3	4	5	6	7	8	9	10
I-го рода	2,82	2,09	1,81	1,26	1,90	2,81	2,80	1,84	2,33	2,32
II-го рода	3,46	2,82	3,19	2,97	2,45	3,24	2,97	2,83	3,52	2,25

Таблица 2.4 – Результаты классификации объектов.

Вид ошибки	Номер испытания									
	1	2	3	4	5	6	7	8	9	10
I-го рода	2,13	2,54	2,89	2,39	2,62	2,32	2,77	2,39	2,96	2,15
II-го рода	2,11	2,39	2,56	2,08	2,60	2,83	2,64	2,86	2,71	2,38

На Рисунке 2.11 представлен вид разработанного аппаратно-программного комплекса системы распознавания препятствий, установленного на автономный трактор.

Рисунок 2.11 – подсистема машинного зрения ИС БСТС

На Рисунках 2.12, 2.13, 2.14 представлены результаты сегментации препятствий подсистемы машинного зрения ИС БСТС.

Рисунок 2.12 – Распознавание столбов

Рисунок 2.13 – Распознавание деревьев и кустарников

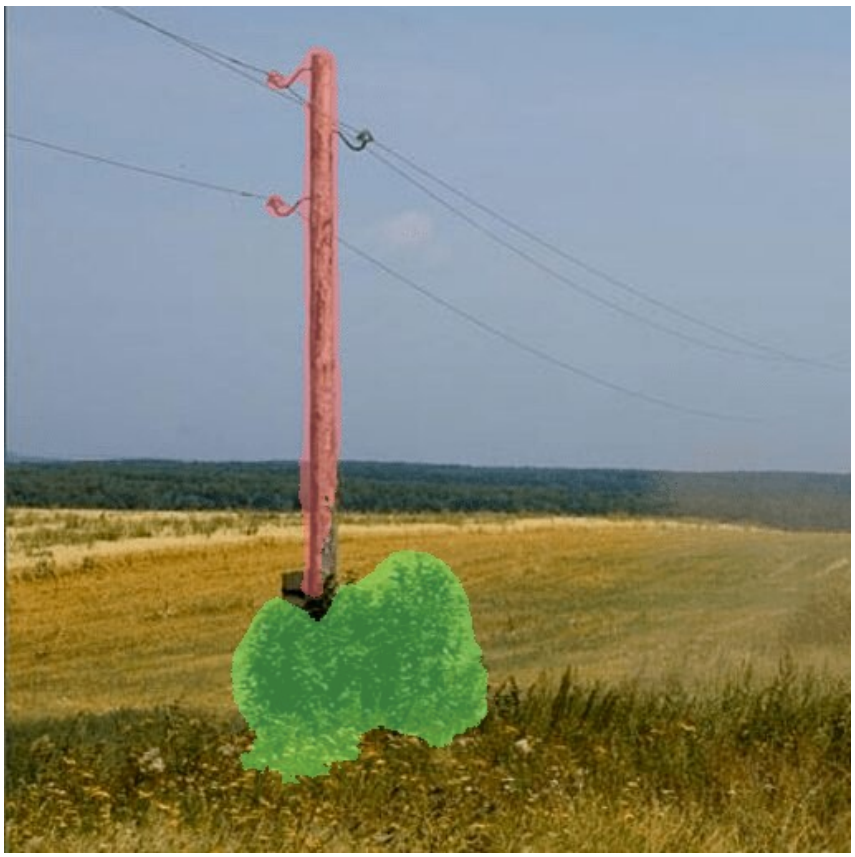


Рисунок 2.14 – Комплексное распознавание столба и куста

Проведенное исследование выявило высокую эффективность применения нейронной сети архитектуры U-Net в задаче сегментации и детектировании - принятия решения по ошибке I-го рода для алгоритмов детектирования составляет в среднем 2,2%, классификации в среднем 2,5%; по ошибке II-го рода для алгоритмов детектирования в среднем 2,9%; классификации в среднем 2,5%.

Таким образом, разработанные алгоритмы нормализации изображений и использование нейронной сети U-Net, совместно с технологией ускорения вычислений на базе векторных ускорителей CUDA, позволили повысить системные ограничения, и обеспечили высокую эффективность работы подсистемы машинного зрения:

1. **Качество обработки изображений**: Алгоритмы оценки уровня шума, коррекции резкости, компенсации засветки и контрастности существенно улучшили качество обработки изображений. Это позволило системе точнее и надежнее распознавать и анализировать разнообразные объекты, такие как

растения, сельскохозяйственная техника и участки, устраняя недочеты, связанные с шумами и низкой резкостью изображений.

2. **Эффективность в условиях неблагоприятных факторов**: алгоритмы нормализации изображений обеспечивают снижение воздействия неблагоприятных факторов, таких как плохая видимость и изменения освещения. Это позволяет подсистеме машинного зрения работать более эффективно и надежно в разнообразных климатических условиях, что особенно важно в сельском хозяйстве.

3. **Скорость обработки данных**: несмотря на внедрение дополнительных алгоритмов нормализации, система, благодаря использованию ускорителей CUDA, не потеряла в скорости обработки данных. Это важно для оперативного анализа информации с камер и датчиков, что обеспечивает более точные и своевременные решения в сельскохозяйственных операциях.

4. **Распознавание разнообразных объектов**: алгоритмы детектирования и классификации объектов, работающие в связке с нейронной сетью U-Net, позволяют системе машинного зрения распознавать и анализировать различные объекты в сельском хозяйстве, включая растения, почву, технику и другие элементы. Это снимает системные ограничения в распознавании разнообразных объектов.

Таким образом, внедрение алгоритмов нормализации, использование нейронной сети U-Net и технологии ускорения вычислений на базе векторных ускорителей CUDA не только повысили системные ограничения, но и значительно усовершенствовали систему, позволяя ей эффективно выполнять сложные задачи распознавания и анализа объектов в условиях сельского хозяйства. Точность, надежность, оперативность и скорость работы системы позволяют повысить производительность и безопасность сельскохозяйственных операций.

Заключение главы 2

Во второй главе исследования, посвященной реализации шаблонов проектирования в инфокоммуникационных системах беспилотных сельскохозяйственных транспортных систем (ИС БСТС), были представлены фундаментальные концепции бережливого производства и теории ограничений Элияху Голдратта. Применяя эти концепции, было рассмотрено, как можно разработать и обеспечить высокое качество таких интеллектуальных систем, что имеет стратегическое значение в контексте сельского хозяйства.

Начав с бережливого производства, сделан вывод, что эффективное использование ресурсов и оптимизация процессов являются ключевыми компонентами успешной разработки ИС БСТС. Далее был переключен фокус с простых производственных структур на сложные, распределенные системы, где ресурсы ограничены и требуют более гибкого и адаптивного подхода.

Далее изучая теорию ограничений, выяснилось, что понимание и управление системными ограничениями играют критическую роль в разработке ИС БСТС. Это не просто ограничения, которые нужно устранить, но и ключевые пункты, которые следует учитывать с самого начала процесса.

Подробно рассмотрев каждую подсистему ИС БСТС, были выявлены конкретные системные ограничения:

1. В вычислительной подсистеме были определены следующие системные ограничения: ограниченность ресурсов и строгая приоритезация задач. Далее успешно преодолены ограниченность ресурсов путем внедрения трехуровневой схемы, оптимизированной для разных уровней задач. Также, системы реального времени сняли строгую приоритезацию задач.

2. В подсистеме коммуникации и связи, системные ограничения включали в себя пропускную способность каналов связи, латентность и задержки, помехоустойчивость, защиту данных и совместимость с разными устройствами. Было показано применение современных технологий, таких как 5G и протокол SCTP, для снятия этих ограничений.

3. В подсистеме позиционирования и навигации, ограничения включали точность позиционирования, доступность сигналов навигационных спутников и зависимость от глобальных систем навигации. Система высокоточной спутниково-инерциальной навигации "Созвездие" успешно сняла эти ограничения, обеспечивая надежное и точное позиционирование.

4. В подсистеме машинного зрения, системные ограничения включали качество обработки изображений, эффективность в условиях неблагоприятных факторов, скорость обработки данных и распознавание разнообразных объектов. Внедрение разработанных алгоритмов нормализации изображений и сверточной нейронной сети U-Net позволило преодолеть эти ограничения.

В итоге, вторая глава исследования представляет собой систематическую работу по определению и преодолению системных ограничений на стадии разработки ИС БСТС. Это подход, который способствует созданию более эффективных, гибких и адаптивных систем, способных эффективно функционировать в разнообразных условиях сельского хозяйства. Следовательно, данная глава является фундаментом для последующей разработки и реализации ИС БСТС, а также для обеспечения их надежности и производительности.

Глава 3. Проектирование и реализация систем автоматизации БСТС.

В мире современной сельской экономики, автоматизация процессов играет ключевую роль в обеспечении эффективности и результативности сельскохозяйственного производства. Одним из средств для достижения этих результатов является автоматизация тракторов. В рамках данного обзора будет представлено описание автоматизации трактора «МТЗ 112Н-01»[70] и «МТЗ-3522 Cummins»[71], подробно рассмотрены его особенности и преимущества.

Внедрение автоматизации в сельское хозяйство – это непростая задача, требующая не только технических инноваций, но и изменения мышления и подходов к работе. Автоматизация тракторов «МТЗ» (Рисунок 3.1) создает возможности для повышения эффективности производства и сокращения затрат на трудовые ресурсы. Однако необходимо учитывать, что успешная реализация автоматизации требует комплексного подхода, включающего разработку и внедрение соответствующих технических решений, а также обучение персонала.

Рисунок 3.1 – Логотип семейства колёсных тракторов «МТЗ»

Дистанционное, а затем и автономное управление будет вводится поэтапно, т.е. от автоматизации минимального набора органов управления, обеспечивающих контролируемое движение трактора (начало движения, движение вперед, возможность поворотов, остановки и окончание движения) до полной автоматизации (все виды движения, полное управление электроникой трактора, управление навесным оборудованием).

При решении задач автоматизации тракторной техники в качестве шаблонов проектирования малого трактора МТЗ-112 и тяжелого трактора

МТЗ-3522 использовались существующие наработки в рассматриваемой предметной области.

Следует отметить, что шаблоны были сформированы как на основе анализа тенденций в области беспилотной техники, так и на основе собственных наработок, полученных коллективом при участии автора в течение реализации ряда проектов по автоматизации различных видов транспортных средств[72-82].

В качестве шаблонов применялись следующие принципы:

1) представление системы управления БСТС в виде многоуровневой иерархической системы;

2) разбиение системы управления БСТС на 4 следующие подсистемы:

- подсистема реконструкции окружающей обстановки (которая включает подсистему сенсорики и систему машинного (компьютерного) зрения);

- подсистема позиционирования и навигации (или подсистема формирования навигационно-временного поля);

- подсистема коммуникации и связи;

- вычислительный контур.

3) принципы формирования требований к подсистемам с учетом их взаимодействия, классификации типов трафика и учетом режимов автоматизированного управления БСТС.

Дополнительно следует отметить, что в приведенных ниже примерах тракторной техники выполнялась автоматизация систем управления серийных образцов. Разрабатывалась так называемые ретрофит решения (от английского retrofit – модернизировать), суть которых – это добавление новой технологии или ее свойств к существующей системе. Такой подход позволяет существенно сократить появление серийных образцов беспилотной сельскохозяйственной техники с высокой степенью автоматизации и соответствует принципам бережливого производства, описанных в предыдущих главах диссертации.

1. Автоматизация трактора «МТЗ-112Н-01»

1.1. Описание трактора «МТЗ-112Н-01»

Мини-трактор «BELARUS 112H-01» является усовершенствованной моделью мини-трактора «BELARUS 132H» и предназначен для выполнения сельскохозяйственных работ на мелкоконтурных земельных участках, выполнения основной и предпосевной обработки почвы, посева и кошения трав, проведения пропашных работ в междурядьях в агрегате с навесными и прицепными машинами и орудиями, выполнения различных работ в коммунальном хозяйстве и промышленности.

Технические характеристики техники указаны в Таблице 3.1.

Рисунок 3.1 – Мини-трактор «BELARUS 112H-01».

Таблица 3.1 – Технические характеристики «BELARUS 112H-01».

Параметр	Значение
Двигатель	«GX390 (Хонда)», бензин
Размер сенсора	570
Число передач вперед/назад	4/3
Агротехнический просвет, мм	295
Колея регулируемая, мм	600, 700, 840
База, мм	1030
Радиус поворота при колее 700 мм, м	2,5
Номинальная мощность, кВт (л.с.)	8,7 (11,8)
Удельный расход топлива г/квт.ч,	313
Скорость движения, км/ч	вперед 2,96 - 18,46. назад 4,03 - 13,47
Габаритные размеры, мм	2500 x 1000 x 2000

1.2. Перечень и описание автоматизируемых систем

Для автоматизации набора органов управления, необходимых для контролируемого движения трактора, были разработаны и установлены блоки управления (Рисунок 3.2 и Рисунок 3.3). Один отвечает за низкоуровневое управление, другой за прием и передачу данных к серверу. Блоки сделаны из металла и закреплены на крыльях.

Блок обеспечивает возможность осуществления следующих функций:

1. Управление асинхронным мотором, приводом задних колес;
2. Поворот колес, с использованием электроуправляемого распределительного гидравлического устройства;
3. Ручное управление трактором с помощью беспроводного контроллера;
4. Распознавание объектов и расстояния до них с помощью камеры;
5. Заблаговременная остановка трактора перед препятствием с помощью лидара;

6. Автопилотирование, автоматический объезд препятствий.

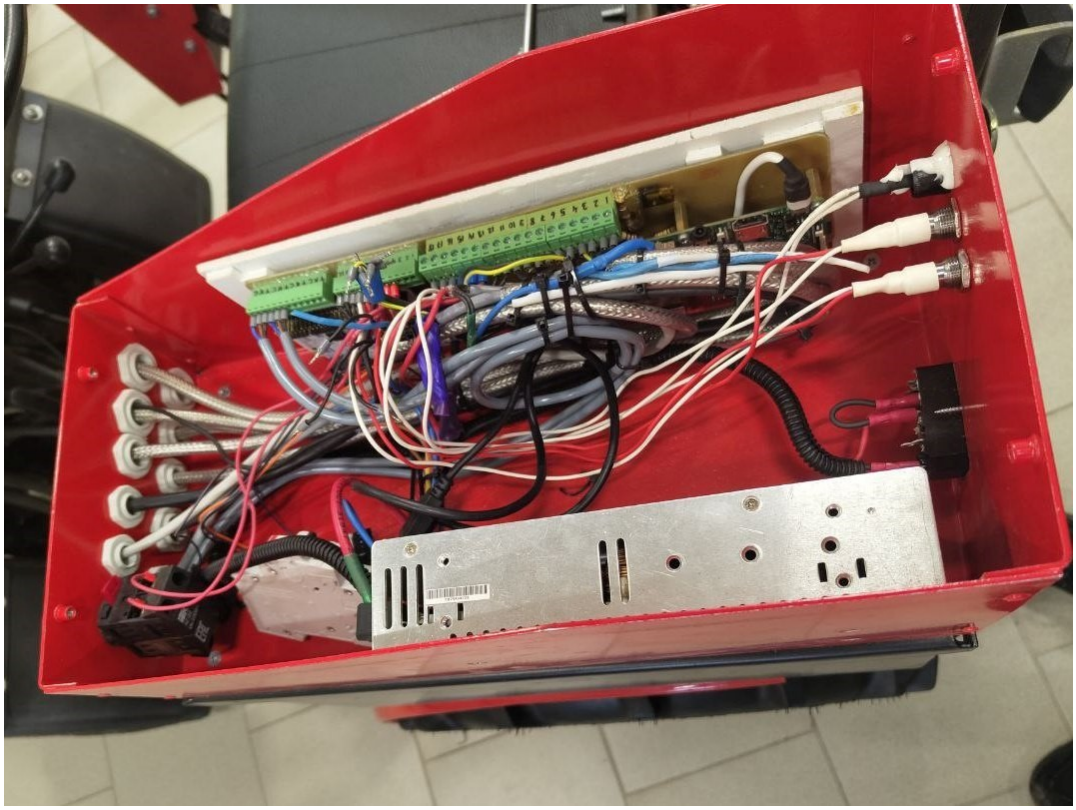


Рисунок 3.2 – Низкоуровневый блок.

Рисунок 3.3 – Блок связи.

Одним из ключевых элементов низкоуровневой части блоков управления является силовая плата управления, оснащенная микроконтроллером на плате типа «NUCLEO-64» с микроконтроллером «STM32». Эта плата отвечает подачу управляющих сигналов к элементам трактора. Принципиальная схема платы приведена в приложении А.

Процесс передачи команд начинается с получения команд от сервера. Сервер отправляет команды через WiFi точку доступа «Ubiquiti Bullet», установленную на тракторе, которая служит связующим звеном между сервером и блоком управления. После этого команды пересылаются по протоколу UART между «Raspberry Pi3» и силовой платой управления.

Преимуществом использования Raspberry Pi3 в качестве клиента является его широкий функционал и возможность обработки и анализа большого объема данных. В свою очередь, использование микроконтроллера STM32 на силовой плате управления позволяет эффективно управлять и контролировать работу различных механизмов и устройств.

В последнее время все больше фермеров и садоводов стремятся к экологически чистым и энергоэффективным решениям. И одним из таких решений стало установление электродвигателя на задний вал трактора (Рисунок 3.4). Такая установка обеспечивает меньше шума и вибраций, что повышает комфорт работы и уровень безопасности.

Для обеспечения питания преобразователя и электродвигателя в трактор были установлены аккумуляторные AGM батареи, обладающие высокой емкостью и напряжением на уровне 64 В (Рисунок 3.5).

Рисунок 3.4 – Электродвигатель, установленный на задний ВОМ. Частотный преобразователь установлен под сидением.

Рисунок 3.5 – Аккумуляторные батареи 64 В.

Для питания логической части установки, такой как электронные схемы, контроллеры и другие устройства, используются отдельные аккумуляторные батареи AGM с напряжением 24 В (Рисунок 3.6). Они обеспечивают энергией все необходимые устройства и гарантируют бесперебойную работу всей системы.

Рисунок 3.6 – Аккумуляторные батареи 24 В.

Для контроля движения трактора на задних колесах были установлены одометры. В данном случае одометр представляет собой систему из диска с равноудаленными метками и бесконтактного индукционного концевика, который читает информацию с меток на диске при их прохождении (Рисунок 3.7). Это позволяет точно определять пройденное расстояние, что важно при выполнении необходимых элементов автопилотирования (ПО YOLO).

Система контроля позволяет обнаруживать и исправлять любые отклонения в движении трактора, например, избыточную скорость, неравномерное движение или смещение по направлению.

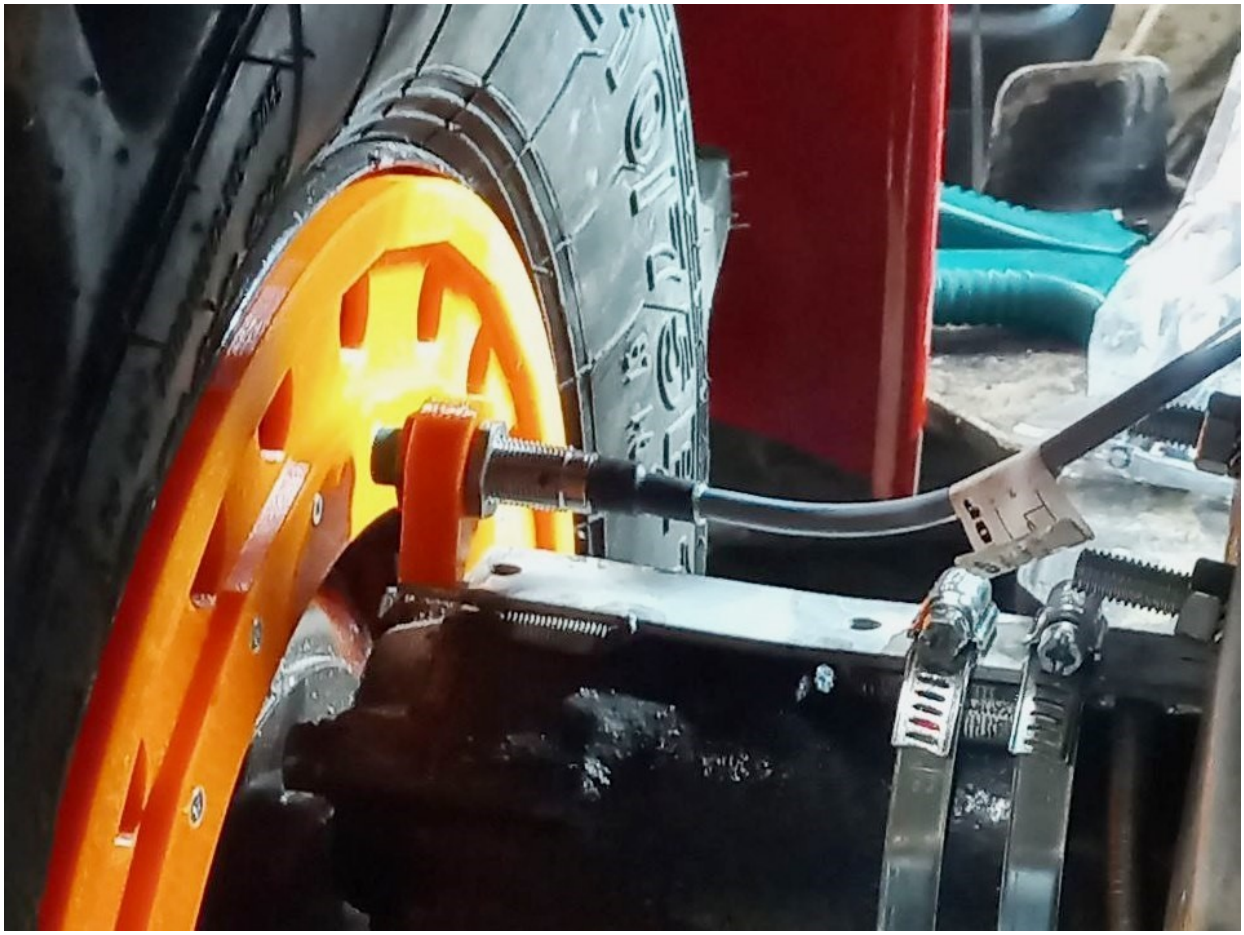


Рисунок 3.7 – Одометр из бесконтактного концевика и колеса изготовленного на 3D принтере.

Для контроля угла поворота такой рамы был установлен потенциометр - устройство, предназначенное для измерения угла поворота объединения сочлененных частей (Рисунок 3.8).

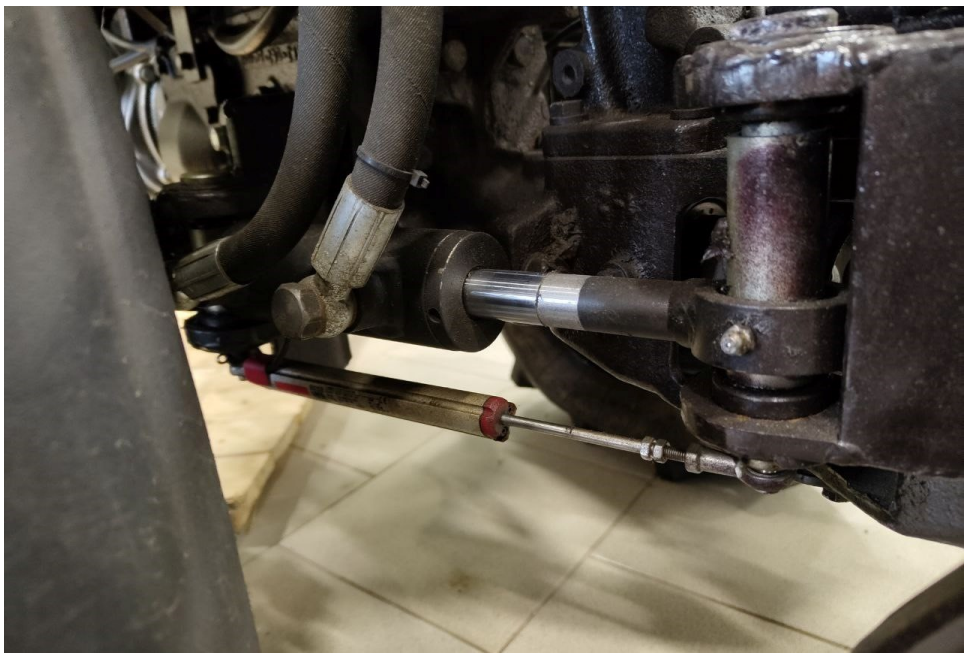


Рисунок 3.8 – Линейный потенциометр, установленный на шарнирно сочлененную раму для контроля угла поворота колес.

Поворот рамой осуществляется использованием электроуправляемого распределительного гидравлического устройства «КОСМОС SK PRD»[83], которое находится под капотом трактора. Это устройство отвечает за управление гидравликой поворота рамы (Рисунок 3.9).

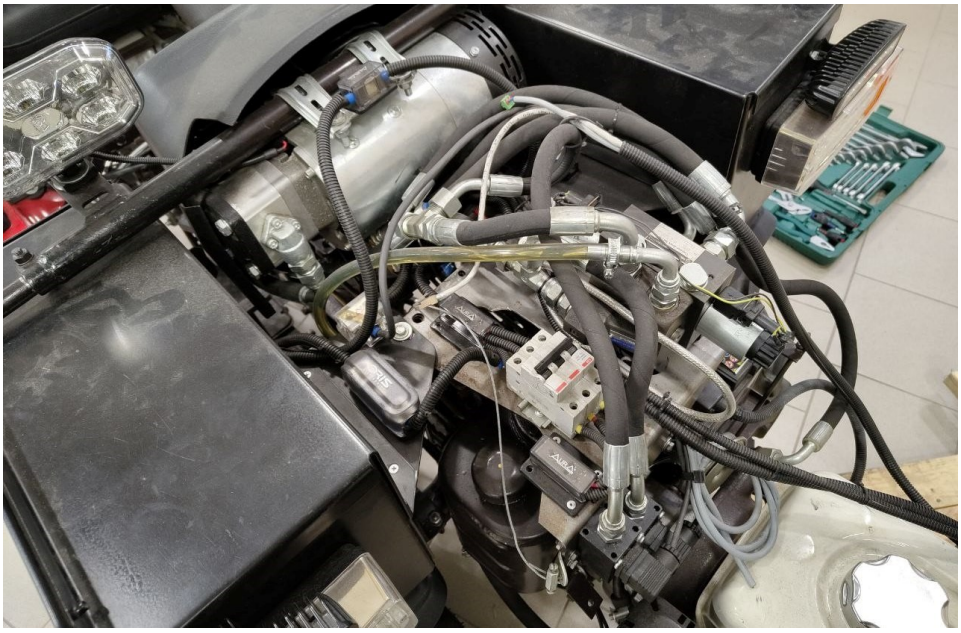


Рисунок 3.9 – Электроуправляемое распределительное гидравлическое устройство «КОСМОС SK PRD» под капотом.

1.3 Используемое ПО

Для обеспечения работы автоматизированного управления трактором используется следующее ПО:

1) Программа для реализации технического зрения сенсорики оптического диапазона системы «Беспилотный трактор КФУ-МТЗ-112»

Программа устанавливается на ПК на базе ОС Windows, и соединяется с планшетом оператора для предоставления информации о местоположении распознанных объектов. Программа не обладает собственным графическим интерфейсом и работает в фоновом режиме.

Представлен следующий функционал:

- на вход по сети поступает информация о распознанных объектах;
- по полученным данным вычисляется относительное местоположение на цифровой карте для каждого объекта;
- полученная информация передается по сети на планшет оператору.

Если при вычислениях устанавливается, что объект находится в радиусе небезопасной зоны, то передается соответствующий сигнал для принятия решения автопилота.

Тип ЭВМ: архитектура x86

Языки: C++

ОС: Windows 10 и новее

Объём программы: 13 Кбайт

2) Программа модификации результатов работы нейросети системы «Беспилотный трактор КФУ-МТЗ-112»

Программа устанавливается на ПК на базе ОС Windows и вычисляет тип и местоположение объектов на снимке. Программа не обладает собственным графическим интерфейсом и работает в фоновом режиме.

Представлен следующий функционал:

- на вход программы по сети поступает видеопоток;
- на полученных изображениях детектируются и классифицируются объекты различных типов;

- после этого данные объектов (тип, местоположение в виде ограничивающего прямоугольника на изображении) передаются по сети для вычисления относительного местоположения найденных объектов.

Тип ЭВМ: архитектура x86

Языки: C++

ОС: Windows 10 и новее

Объём программы: 171 Кбайт

3) Программа низкоуровневого управления системы «Беспилотный трактор КФУ-МТЗ-112»[84]

Программа, устанавливаемая на микроконтроллер серии STM32 обеспечивает прием командных пакетов через встроенный USB-Serial адаптер, отправку данных состояния с опрашиваемых датчиков и данных самодиагностики управления и питания. Обеспечивает управление, в соответствии с полученными командами, электромеханическими частями трактора, и, в соответствии с данными самодиагностики, внешними индикаторами.

Представлен следующий функционал: опрос и фильтрация значений датчика поворота руля; опрос датчиков положения тормоза и заряда аккумуляторов; опрос одометра и вычисление угловой скорости поворота колеса; опрос состояния аварийной кнопки торможения; анализ состояния управляющих сигналов и датчиков положения механического тормоза и отработка их сбросов при наличии недопустимых значений; прием команд и отправка пакета состояния трактора через USB-Serial адаптер; расчет угла и управление поворотом руля; управление направлением (вперед, назад, нейтраль) и скоростью движение трактора; управление механическим тормозом трактора; управление внешними индикаторами (индикаторы заряда аккумуляторов, индикация приема команд с USB-Serial адаптера); отработка аварийной остановки при нажатии кнопки аварийной остановки.

Тип ЭВМ: STM32f303re

Языки: C

ОС: Отсутствует

Объём программы: 57 Кбайт

4) Программа низкоуровневой обработки данных сенсорики лидарного типа системы «Беспилотный трактор КФУ-МТЗ-112»[85]

Программа, устанавливаемая на микроконтроллер серии STM32 обеспечивает конфигурацию отправки данных и прием данных с лидара по интерфейсу CAN и отправку этих данных по интерфейсу Ethernet на приемник данных.

Представлен следующий функционал: прием данных с лидара; отправка конфигурационного пакета на лидар; формирования и отправка пакета на приемник данных.

Тип ЭВМ: STM32f407

Языки: C

ОС: FreeRTOS

Объём программы: 239 Кбайт

5) Программа низкоуровневой обработки сигналов системы «Беспилотный трактор КФУ-МТЗ-112»

Программа, устанавливаемая на мини-компьютер Raspberry Pi обеспечивает трактору беспроводную связь платы управления трактором с компьютера оператора. Мини-компьютер общается с платой управления по проводной связи USB, с компьютером оператора по беспроводному интерфейсу WiFi.

Представлен следующий функционал: прием данных состояния трактора с платы управления; фильтрация битых пакетов состояния; ограничение размера приема данных состояния; отправка проверенных пакетов состояния в компьютер оператора трактора; прием командных пакетов с компьютера оператора трактора; фильтрация битых командных пакетов; ограничение размера командных пакетов; отправка проверенных командных пакетов на плату управления трактором.

Тип ЭВМ: микроархитектура Cortex-A53

Языки: Python 2.5

ОС: Raspbian os

Объём программы: 4 Кбайт

6) Программа для централизованного приёма-передачи и контроля программных пакетов системы «Беспилотный трактор КФУ-МТЗ-112»
Программа функционирует на ПК, обеспечивает взаимодействие и связь между отдельными программами и дистанционное управление трактором с джойстика.

Представлен следующий функционал: прием и обработка пакетов от «Графическая программа оператора для дистанционного управления ТС системы «Беспилотный трактор КФУ-МТЗ-112»», таких как: пакеты управления трактором, пакет, включения или выключения режима обработки данных с лидара, пакет, включения или выключения режима обработки данных с камеры; прием и обработка сигналов управления, сигнала паузы, сигнала стоп, сигнала движения по кругу, сигнала движения по восьмерке от джойстика Xbox 360; прием и обработка пакетов управления трактором от «Программа для реализации подсистемы автономного управления системы «Беспилотный трактор КФУ-МТЗ-112»»; прием и обработка сигнала тревоги от «Программа для реализации технического зрения лидарного типа сенсорики системы «Беспилотный трактор КФУ-МТЗ-112»»; прием и обработка результатов детектирования объектов, а также сигнала тревоги от «Программа для реализации технического зрения сенсорики оптического диапазона системы «Беспилотный трактор КФУ-МТЗ-112»»; пересылка пакетов состояния трактора от «Программа низкоуровневой обработки сигналов системы «Беспилотный трактор КФУ-МТЗ-112»» на «Программа для реализации подсистемы автономного управления системы «Беспилотный трактор КФУ-МТЗ-112»» и «Графическая программа оператора для дистанционного управления ТС системы «Беспилотный трактор КФУ-МТЗ-

112»»; отсылка пакетов управления трактором на «Программа низкоуровневой обработки сигналов системы «Беспилотный трактор КФУ-МТЗ-112»».

Тип ЭВМ: архитектура x86

Языки: Python

ОС: Windows 10 и новее

Объём программы: 30223 байт

7) Программа для реализации подсистемы автономного управления системы «Беспилотный трактор КФУ-МТЗ-112»

Программа функционирует на ПК и взаимодействует с трактором через «Программа для централизованного приёма-передачи и контроля программных пакетов системы «Беспилотный трактор КФУ-МТЗ-112»». Программа является консольной, информация о текущем состоянии трактора отображается в консоли.

Представлен следующий функционал: режим автопилота «движение по кругу», режим автопилота «движение по восьмерке», построение теоретического наиболее безопасного и наикратчайшего пути от начального положения трактора к цели на определенной карте с определенным множеством препятствий, движение по пути близкому к теоретическому с учетом возможностей и габаритов трактора от начального положения трактора к цели на определенной площадке с определенным множеством препятствий.

Тип ЭВМ: архитектура x86

Языки: MatlabR2019a

ОС: Windows 10 и новее

Объём программы: 28907 байт

8) Графическая программа оператора для управления подсистемой автономного управления системы «Беспилотный трактор КФУ-МТЗ-112»

Программа функционирует на планшете оператора на базе ОС Windows и позволяет управлять беспилотным трактором. Программа обладает собственным графическим интерфейсом, который разработан под разрешение экрана 1440x960px.

Представлен следующий функционал: загрузка цифровой карты текущей местности, на которой расставлены статические объекты; установка начального местоположения беспилотного трактора; установка начального направления беспилотного трактора; установка на карте ключевых точек, по котором должен проехать беспилотный трактор; запуск проезда беспилотного трактора; отображение текущего местоположения беспилотного трактора и информации о пройденном пути.

Тип ЭВМ: архитектура x86
Языки: C++
ОС: Windows 10 и новее
Объём программы: 165 Кбайт

9) Графическая программа оператора для создания цифровой карты местности системы «Беспилотный трактор КФУ-МТЗ-112»

Программа функционирует на планшете оператора на базе ОС Windows и позволяет создавать цифровые карты местности. Программа обладает собственным графическим интерфейсом, который разработан под разрешение экрана 1440x960px.

Представлен следующий функционал: создание цифровой карты местности; загрузка изображения местности; задание размеров местности; расстановка статических объектов на местности; сохранение цифровой карты местности.

Тип ЭВМ: архитектура x86
Языки: C++

ОС: Windows 10 и новее

Объём программы: 53 Кбайт

10) Графическая программа оператора для дистанционного управления ТС системы «Беспилотный трактор КФУ-МТЗ-112»

Графическое программа функционирует на ПК, отрисовывает графический интерфейс, с которым взаимодействует оператор для дистанционного управления транспортным средством. Графический интерфейс программы является динамическим и поддерживает устройства с разрешениями экранов от 960x1440 px до 4096×3072 px.

Представлен следующий функционал: передача пакетов управления при взаимодействия оператора с графическим интерфейсом (управление тормозом, управление направлением и скоростью движения, управление углом поворота руля, управление остановкой по машинному зрению и/или по данным с лидара, управление выравниванием руля, управление сброса скорости); прием пакетов обратной связи, содержащие заряд аккумуляторов, состояние тормоза, скорость движения, повороты руля, направление движения; прием данных RTSP потока с камеры и отображение видео; прием пакетов обратной связи, содержащие координаты распознанных объектов и тревоги; отрисовка рамок распознавания в соответствии с полученными координатами; индикация заряда аккумуляторов; индикация включения и выключения остановки перед пешеходом по данным с машинного зрения и/или по данным с лидара.

Тип ЭВМ: архитектура x86

Языки: Python 3.10

ОС: Windows 10 и новее

Объём программы: 79 Кбайт

11) Программа для калибровки подсистемы технического зрения системы «Беспилотный трактор КФУ-МТЗ-112»

Программа функционирует на ПК и служит для калибровки формулы расстояния до объекта от его высоты на изображении для произвольной линзы камеры. Формула нужна для «Программа для реализации технического зрения сенсорики оптического диапазона системы «Беспилотный трактор КФУ-МТЗ-112».

Для объекта определенной высоты на вход программе подаются множество пар расстояний до объекта и высоты объекта на изображении. По этим данным для определенной заранее модели формулы вычисляются неизвестные коэффициенты.

Тип ЭВМ: архитектура x86
Языки: MatlabR2019a
ОС: Windows 10 и новее
Объём программы: 6200 байт

12) Программа для реализации технического зрения лидарного типа сенсорики системы «Беспилотный трактор КФУ-МТЗ-112»

Программа функционирует на ПК и взаимодействует с трактором через «Программа для централизованного приёма-передачи и контроля программных пакетов системы «Беспилотный трактор КФУ-МТЗ-112»». Программа является консольной, получаемые от лидара расстояния для каждого лучевого сегмента отображаются на консоли.

Представлен следующий функционал: прием информации от лидара, отправка пакета остановки при появлении в небезопасной зоне препятствия, отправка разрешающего движения пакета при отсутствии препятствий в небезопасной зоне, возможность изменять размеры небезопасной зоны.

Тип ЭВМ: архитектура x86

Языки: MatlabR2019a

ОС: Windows 10 и новее

Объём программы: 3427 байт

13) Программа для реализации стендовой эмуляции локационных данных системы «Беспилотный трактор КФУ-МТЗ-112»

Программа устанавливается на ПК на базе ОС Windows. Программа позволяет эмулировать локационные данные беспилотного трактора. Программа обладает собственным графическим интерфейсом.

Представлен следующий функционал: эмуляция локационных данных; отправка сгенерированных данных по сети.

Тип ЭВМ: архитектура x86

Языки: C++

ОС: Windows 10 и новее

Объём программы: 17 Кбайт

Демонстрационные сценарии:

1) Управление по геймпаду;

2) Трактор едет по кругу;

3) Трактор едет по восьмерке;

4) Трактор едет по карте (ставим трактор и одну точку маршрута так, чтобы между ними по прямой было препятствие, которое надо объехать).

Типы распознаваемых объектов машинным зрением:

1) Люди;

2) Авто.

Трактор управляется как с джойстика, так и с сервера в виде ПК (Рисунок 3.10).

Рисунок 3.10 – Общий вид трактора.

2. Автоматизация трактора «МТЗ-3522»

2.1. Описание трактора «МТЗ-112Н-01»

Трактор предназначен для выполнения энергоемких сельскохозяйственных работ в тяговом и тяговоприводном режимах в составе широкозахватных и комбинированных агрегатов, в том числе при эшелонированной навеске, на основной и предпосевной обработке почвы, посеве зерновых и других культур, заготовке кормов, уборке корнеплодов, зерновых и технических культур; для выполнения транспортных работ, стационарных работ, в строительстве и промышленности (Рисунок 3.11). Особенностью трактора является гидромеханическая коробка передач с электрогидравлическим управлением.

Технические характеристики техники указаны в Таблице 3.2.

Рисунок 3.11 – «МТЗ-3522 Cummins» в варианте со сдвоенными колесами.

Таблица 3.2 – Технические характеристики «МТЗ-3522 Cummins».

Двигатель	
Двигатель	QSL8.9-C360-III (Cummins)
Мощность двигателя, кВт (л.с.)	264 (359)
Номинальная частота вращения вала, об/мин	2100
Число цилиндров, шт.	6
Рабочий объём, л	8,9
Максимальный крутящий момент, Нм	1500
Удельный расход топлива, г/кВтч	229
Коэффициент запаса крутящего момента, %	25

Ёмкость топливного бака, л	650
Трансмиссия	
Муфта сцепления	сухая, двухдисковая
Коробка передач	гидромеханическая
Переключение передач	электрогидравлическое
Число передач: вперёд/назад	36/24 с ходоуменьшителем
Скорость движения: вперёд/назад	0,34-40/0,43-20,8
ВОМ задний:	независимый, двухскоростной
Макс. мощность на переднем ВОМ, кВт	(л. с.) 58 (79)
Макс. мощность на заднем ВОМ, кВт (л. с.)	219 (298)
Управление ВОМ	гидравлическое
Блокировка дифференциала	гидравлическое
Размеры и масса	
Длина общая (с балластом ПНУ), мм	6500
Ширина, мм	2740
Высота по кабине, мм	3350
База трактора, мм	3000
Колея, мм: по передним колесам	2000, 2150
по задним колесам	2130-2220, 2316-2636
Агротехнический просвет, мм	450
Наименьший радиус поворота, м	6,5
Масса эксплуатационная, кг	12300

В составе этой техники имеется универсальная гидросистема управления рабочими органами сельскохозяйственных орудий с насосом переменной производительности, распределителем с электронным управлением и возможностью программирования последовательности включения/выключения секций, величины расхода, времени действия посредством считывания сигналов с двух датчиков: с силовым, позиционным и смешанным способами регулирования глубины обработки почвы, 4 пары независимых выводов (технические характеристики указаны в Таблице 3.3).

Таблица 3.2 – Технические характеристики гидронавесной системы «МТЗ-3522 Cummins».

Гидронавесная система	
Максимальная грузоподъемность на оси подвеса, кг:	ПНУ/ЗНУ 5000/10000
Максимальное давление, МПа	20
Производительность насоса, л/мин.	0-160
Ёмкость гидросистемы, л	100

2.2. Перечень и описание автоматизируемых систем

Для автоматизации набора органов управления, необходимых для контролируемого движения трактора, был разработан и установлен блок управления (Рисунок 3.12).

На данный момент этот блок обеспечивает возможность осуществления следующих функций:

1. Цифровое управление коробкой передач;
2. Установка переднего или заднего хода;
3. Управление педалями тормоза и сцепления с помощью актуаторов;
4. Цифровое управление педалей газа;
5. Поворот колес, используя гидроблок;

6. Управление трактором с помощью беспроводного контроллера.

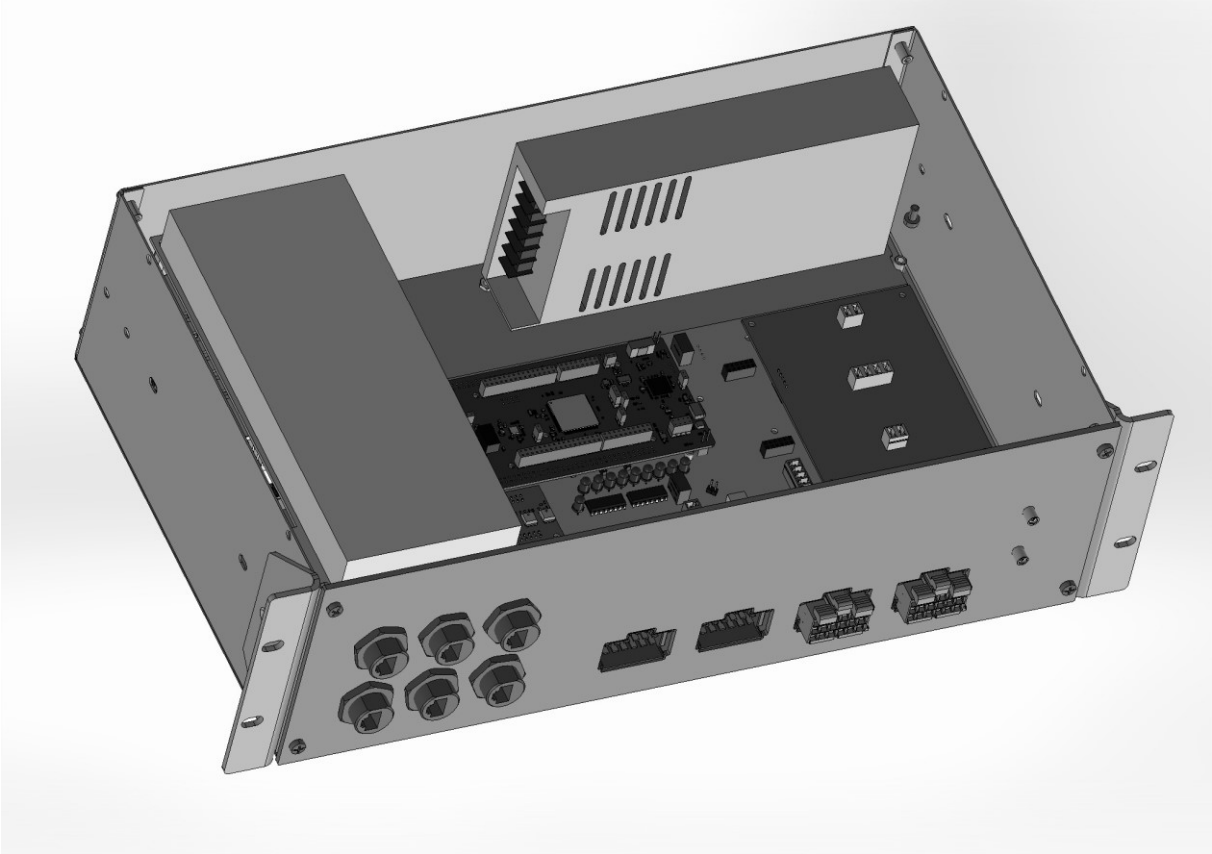


Рисунок 3.12 – 3D модель блока управления трактором.

Управление коробкой передач осуществляется подачей цифровых сигналов на колодку XS6 (Рисунок 3.13). Выбор переднего или заднего хода осуществляется подачей силового сигнала на колодку рычага переключателя хода.

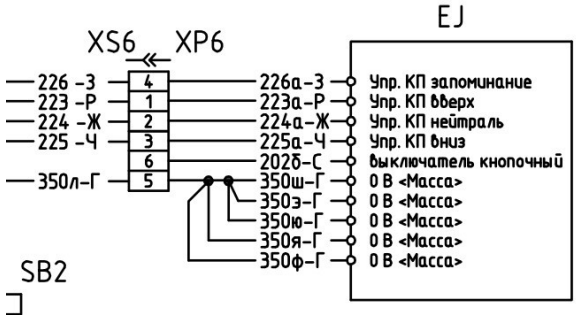


Рисунок 3.13 – Справа часть схемы «Электрические соединения комплексной системы управления БД, ПВМ, ВОМ и переключением

передач трактора Беларус-3522/3525» (см. приложение Б). Слева EJ - джойстик переключения передач.

Подача газа осуществляется с помощью аналогового сигнала через ЦАП, который подается на 2 контакт колодки XS6.1 (Рисунок 3.14). Это позволяет регулировать количество подаваемого топлива и обеспечивать плавное ускорение трактора.

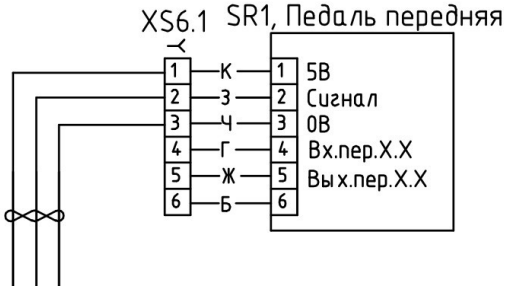


Рисунок 3.14 – Часть схемы «Электрические соединения электронной системой управления двигателем трактора Беларус-3522/3525 Cummins» (см. приложение В).

Педаль сцепления и тормоза нажимается с помощью тяги и актуаторов. Блок управления координирует эти действия с помощью бесконтактных концевиков, обеспечивая точное управление сцеплением и торможением.

Поворот колес осуществляется подачей ШИМ сигнала на соленоиды гидроблока, который встроен в гидравлическую систему трактора (Рисунок 3.15). Это позволяет точно и быстро поворачивать передние колеса трактора.

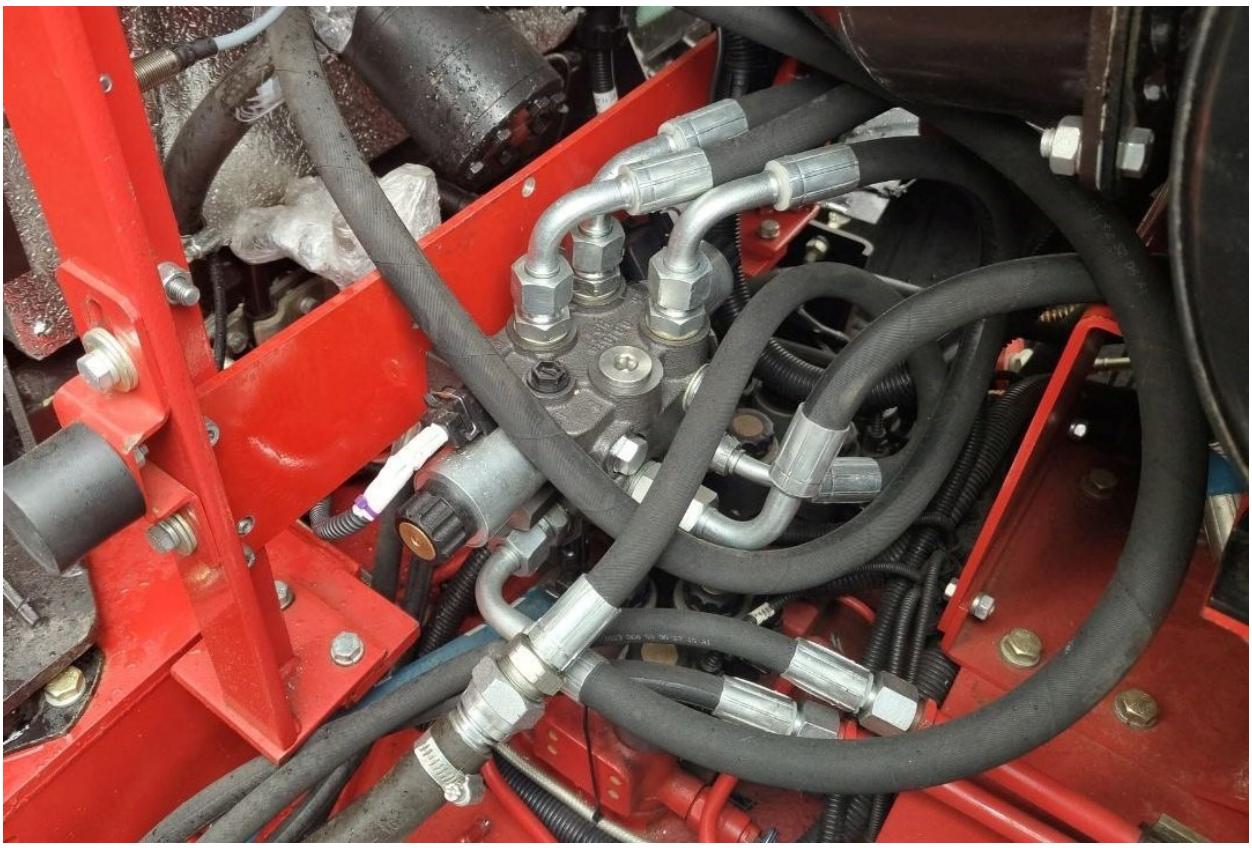


Рисунок 3.15 – Электроуправляемое распределительное гидравлическое устройство «КОСМОС SK PRD», вмонтированное в гидравлическую систему трактора.

2.3. Принцип автоматизации (используемое оборудование, материалы)

Одной из основных задач блока управления является обеспечение плавного и безопасного старта движения трактора и последующей его остановки. Благодаря специальному программному обеспечению, блок управления контролирует процесс запуска алгоритма переключения передач, скорости оборотов двигателя и передает мощность на приводные колеса, плавно отпуская педаль сцепления, позволяя трактору начать движение без рывков (Рисунок 3.16).

Одной из ключевых функций блока управления является обеспечение возможности поворотов. Благодаря потенциометру, установленному на рулевую рейку (Рисунок 3.17), блок управления определяет угол поворота и автоматически регулирует работу колес, обеспечивая плавные и точные

повороты трактора. Потенциометр является временным решением, планируется установка абсолютного энкодера с CAN-шиной, разработаны чертежи и 3D модели (Рисунок 3.18).

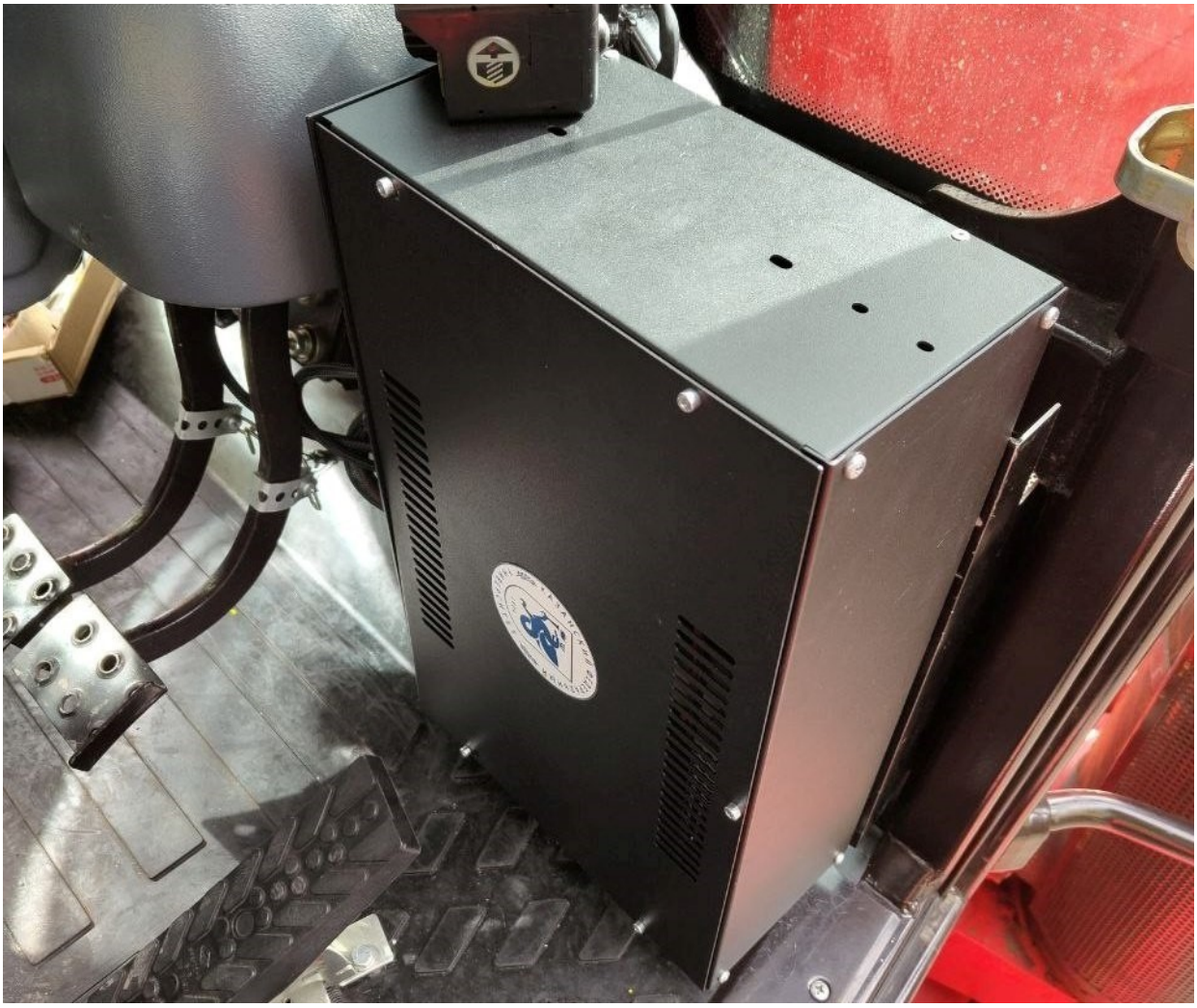


Рисунок 3.16 – Блок управления, установленный в кабину трактора.

Рисунок 3.17 – Линейный потенциометр.

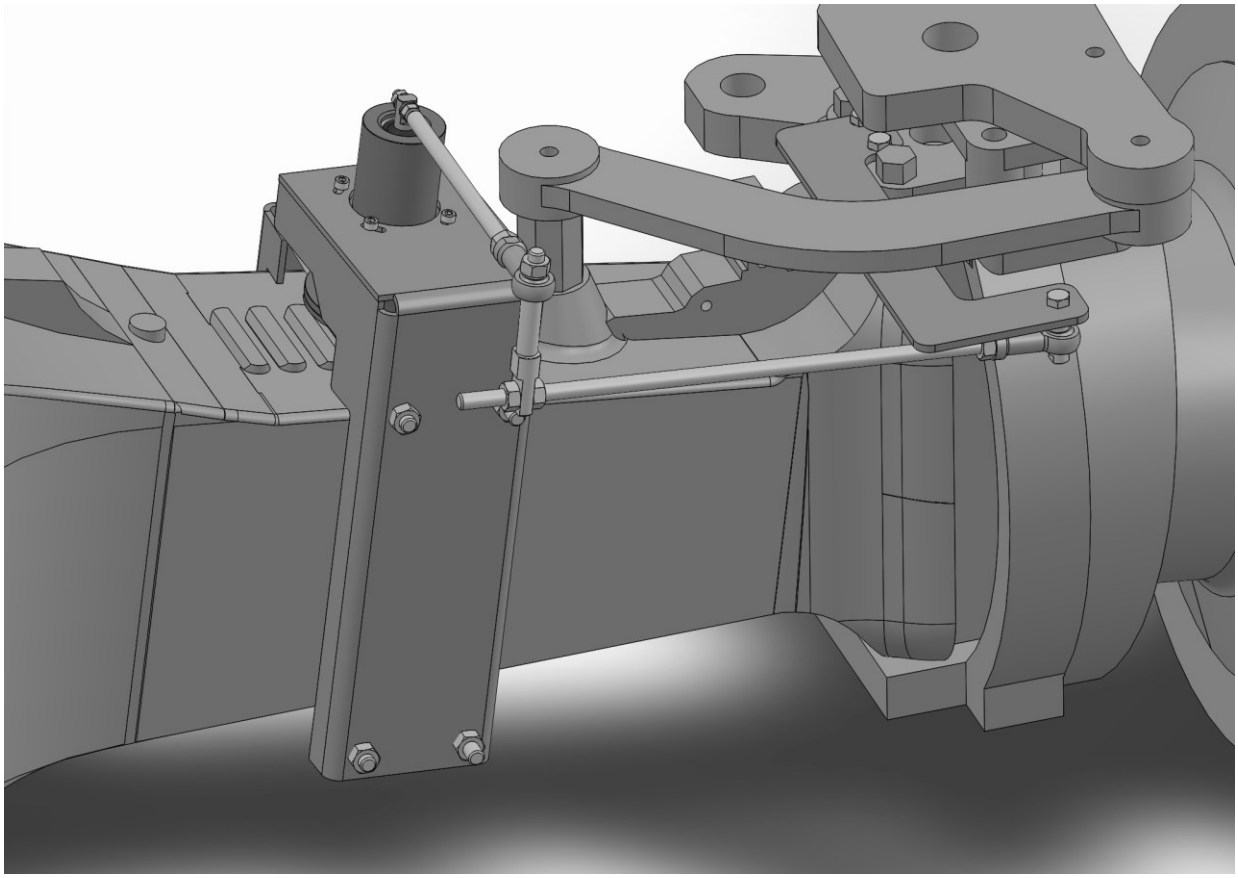


Рисунок 3.18 – 3D модель крепления.

Блок управления (Рисунок 3.19) состоит из двух частей: высокоуровневой части, включающей в себя микрокомпьютер «NVIDIA Jetson» и низкоуровневой части, включающей логическую плату типа «NUCLEO-144» с микроконтроллером «ARM STM32» установленную в силовую плату. Принципиальная схема платы блока управления приведена в приложении Г.

Главная идея состоит в том, что высокоуровневый компьютер, оснащенный мощным процессором «NVIDIA Jetson», отвечает за обработку и анализ данных, а также принятие решений по управлению трактором. В то же время низкоуровневая плата «NUCLEO-144», оснащенная микроконтроллером, отвечает за отправку управляющих сигналов физическим компонентам трактора.

Соединение между компьютером и микроконтроллером осуществляется с помощью интерфейса UART, позволяющего передавать данные между устройствами. Однако, в дальнейшем планируется использование протокола Ethernet для более надежного и быстрого соединения. Принимая команды от высокоуровневого компьютера, микроконтроллер отправляет управляющие сигналы через один из четырех 18-контактных разъемов на блоке. Это позволяет контролировать различные компоненты трактора, такие как двигатель, коробка передач и т.д.

Особенностью данной системы является полная гальваническая развязка между электроникой трактора и блоком управления. Такая развязка важна для обеспечения безопасности работы системы и защиты от возможных электрических повреждений.

Питание для блока также реализовано с помощью гальванически изолированного стабилизатора, который берет энергию от бортовой сети трактора напряжением 12 В. Для обеспечения защиты от короткого замыкания и перегрузки, предусмотрен предохранитель на 20 А.

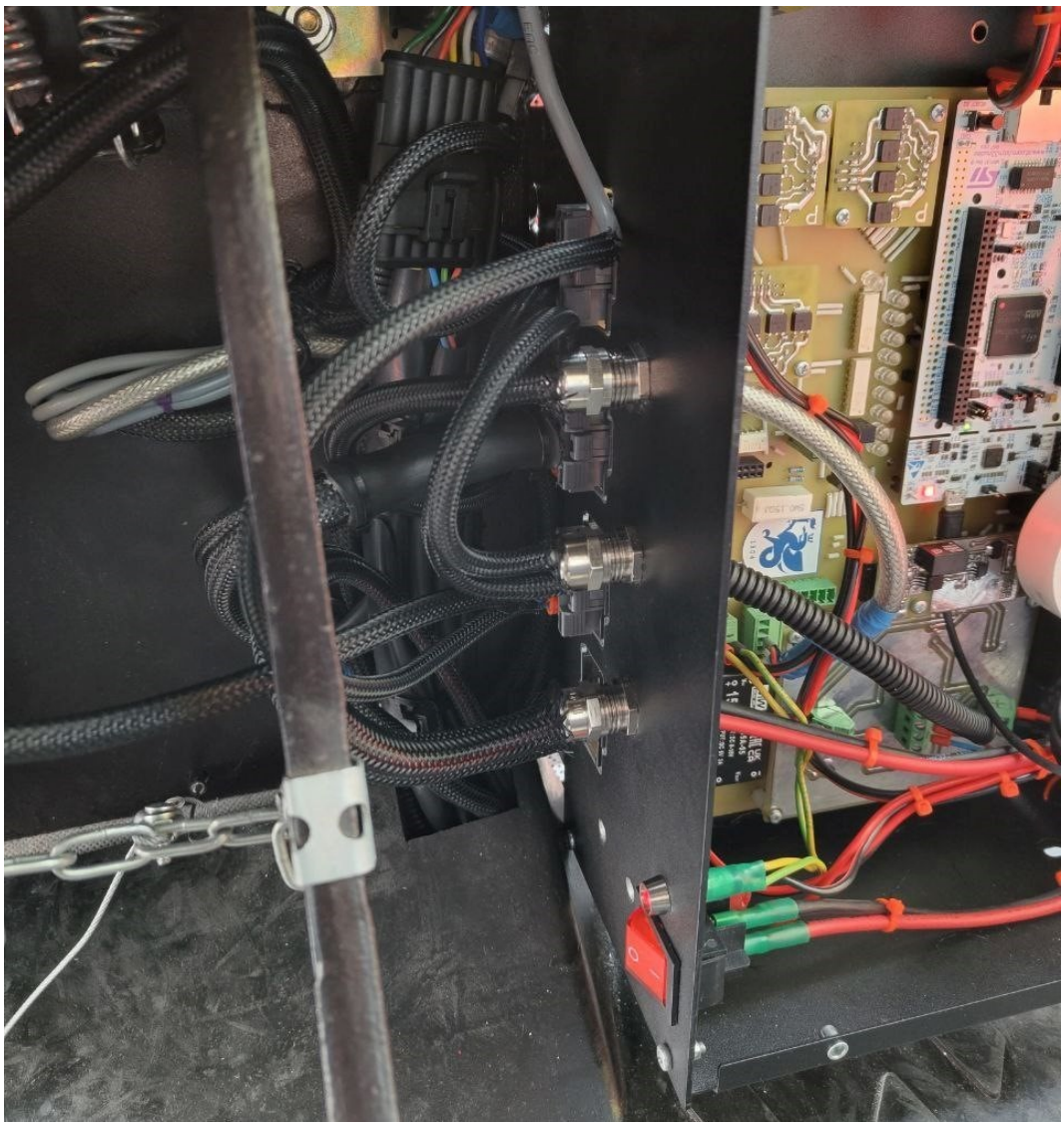


Рисунок 3.19 – Блок управления.

На левой стороне имеется четыре 18-контактных разъема для подключения к трактору. Также видны три ввода в блок: для актуаторов, для питания и для потенциометра. Провода подключения незаметно проложены через обшивку трактора.

2.4. Управление трактором

Трактор управляется с помощью беспроводного джойстика, команды от него принимает микрокомпьютер (Рисунок 3.20). Управление газом осуществляется с помощью левого стика, который позволяет оператору плавно регулировать скорость движения трактора. Оттягивая стик вверх, оператор увеличивает скорость движения вперед, а оттягивая его вниз -

включает движение назад. В то же время, правый стик отвечает за управление рулевым колесом. Поворачивая стик влево или вправо, оператор может осуществлять развороты и повороты. Такой способ управления дает оператору полный контроль над направлением движения трактора, что особенно важно при работе в тесных условиях или на ограниченных площадях.

Помимо основных функций управления газом и рулем, джойстик также оснащен кнопками, которые отвечают за педали и коробку передач. Благодаря этим кнопкам, оператор может легко переключаться между передачами и использовать педали тормоза и сцепления.

Рисунок 3.20 – Беспроводной контроллер, с помощью которого осуществляется управление трактором.

2.5. Применяемое программное обеспечение

Для обеспечения работы автоматизированного управления трактором используется следующее ПО:

1) Программа для реализации графического интерфейса оператора для контроля параметров и запуска режимного управления беспилотным трактором тяжелого класса.

Программа предназначена для контроля параметров и запуска режимного управления беспилотным трактором тяжелого класса, содержит графический интерфейс, с которым взаимодействует оператор для дистанционного управления транспортным средством.

Представлен следующий функционал:

- передача пакетов управления при взаимодействии оператора с графическим интерфейсом (тормозом, направлением и скоростью движения, углом поворота руля, остановкой по машинному зрению и/или по данным с лидара, выравниванием руля, сбросом скорости);

-прием пакетов обратной связи, содержащие заряд аккумуляторов, состояние тормоза, скорость движения, повороты руля, направление движения; прием данных RTSP потока с камеры и отображение видео;

- прием пакетов обратной связи, содержащие координаты распознанных объектов и тревоги;

- отрисовка рамок распознавания в соответствии с полученными координатами; индикация заряда аккумуляторов; индикация включения и выключения остановки перед пешеходом по данным с машинного зрения и/или по данным с лидара.

Тип ЭВМ: архитектура x86

Языки: Python

ОС: Windows 10 и новее

Объём программы: 217 Кбайт

2) Программа построения модели цифрового двойника беспилотного трактора тяжелого класса для системы виртуального моделирования.

Программа предназначена для построения модели цифрового двойника беспилотного трактора тяжелого класса КФУ-МТЗ-5222 для проведения виртуальных испытаний.

Представлен следующий функционал:

- построение 3D модели трактора тяжелого класса КФУ-МТЗ-5222 по заданным параметрам;
- осуществление эмуляции кинематической модели трактора тяжелого класса КФУ-МТЗ-5222.

Программа устанавливается на ПК на базе ОС Ubuntu, и подгружается в качестве подпрограммы в «Программа системы виртуального моделирования испытательного комплекса

Тип ЭВМ: архитектура x86

Языки: XML

ОС: Ubuntu 18.04 и новее

Объём программы: 7566 Кбайт

3) Программа реализации системы виртуального моделирования испытательного комплекса беспилотного трактора тяжелого класса.

Программа предназначена для реализации системы виртуального моделирования испытательного комплекса беспилотного трактора тяжелого класса.

Представлен следующий функционал:

– эмуляция движения транспортного средства;

– эмуляция поведения транспортного средства, при взаимодействии с системой детектирования, классификации и определения положения объектов в оптическом диапазоне.

– эмуляция поведения транспортного средства при взаимодействии с системой автономного управления.

Программа устанавливается на персональный компьютер на базе ОС Ubuntu, и является подпрограммой «Программы построения модели цифрового двойника беспилотного трактора тяжелого класса для системы виртуального моделирования».

Тип ЭВМ: архитектура x86
Языки: Python
ОС: Ubuntu 18.04 и новее
Объём программы: 15 Кбайт

4) Программа автоматического объезда поля для беспилотных тракторов тяжелого класса

Программа предназначена для определения оптимальной траектории объезда поля для беспилотных тракторов тяжелого класса.

В программе реализован следующий функционал:
- ввод данных размера поля, координат местонахождения трактора;
- построение теоретического наиболее безопасного и наикратчайшего пути от начального положения трактора к цели на определенной карте с целью объезда указанного участка местности для его максимального покрытия;
- выполнение команд движения согласно выбранному сценарию.

Программа функционирует на персональном компьютере и взаимодействует с трактором через подсистему автономного управления трактором.

Программа является консольной, информация о текущем состоянии трактора отображается в консоли.

Тип ЭВМ: архитектура x86
Языки: Python

ОС: Windows 10 и новее

Объём программы: 115 Кбайт

5) Программа автоматизации управления беспилотным трактором тяжелого класса для достижения контрольных пунктов на карте.

Программа предназначена для автоматизации процесса управления беспилотным трактором тяжелого класса по заданным на карте координатам (контрольным точкам).

Представлен следующий функционал: режим автопилота «контрольные пункты», построение теоретического наиболее безопасного и оптимального пути от начального положения трактора к контрольным точкам на определенной карте местности с учетом статических и динамических препятствий, реагирование на появление пешехода в зоне работы трактора.

Программа функционирует на персональном компьютере и взаимодействует с трактором через подсистему автономного управления трактором. Программа является консольной, информация о текущем состоянии трактора отображается в консоли.

Тип ЭВМ: архитектура x86

Языки: Python

ОС: Windows 10 и новее

Объём программы: 119 Кбайт

6) Программа детектирования, классификации и определения положения объектов в оптическом диапазоне для беспилотного трактора тяжелого класса.

Программа вычисляет тип и местоположение объектов на снимке в оптическом диапазоне беспилотного трактора тяжелого класса.

Представлен следующий функционал:

– на вход программы по сети поступает видеопоток;

– на изображения видеопотока детектируются и классифицируются объекты различных типов;

– полученные данные (тип, местоположение в виде ограничивающего прямоугольника на изображении) передаются по сети для вычисления относительного местоположения найденных объектов.

Программа устанавливается на персональном компьютере на базе ОС Ubuntu и не обладает собственным графическим интерфейсом и работает в фоновом режиме.

Тип ЭВМ: архитектура x86
Языки: C++
ОС: Ubuntu 18.04 и новее
Объём программы: 28 Кбайт

2.6. Перспективы дальнейшей автоматизации

В блоке управления для автоматизации трактора присутствует множество функциональных возможностей, хотя в настоящее время они еще не используются. Один из таких компонентов этого блока - две шины CAN, предназначенные для подключения различной периферии. Эти шины могут использоваться для подключения парктроников, радаров или лидаров.

Блок управления оборудован двумя блоками GNSS, позволяющими получать точные данные о местоположении трактора с использованием спутниковых систем. Эти данные могут быть использованы для различных целей, например, для определения абсолютной ориентации трактора.

Также блок управления имеет 16 сухих контактов, которые могут использоваться для управления различными устройствами трактора. Например, с помощью этих контактов можно управлять фарами, маячками, навесным оборудованием или подключать одометры ВОМ. Блок управления также предоставляет 4 контакта для подключения дополнительных

концевиков, которые могут быть использованы для контроля положения различных частей дополнительных механизмов.

Кроме того, блок управления оборудован 10 резервными универсальными портами ввода-вывода. Они могут быть коммутированы через дополнительную плату расширения, что позволяет увеличить функциональность блока управления и подключить дополнительные устройства или датчики.

Блок имеет 6 Ethernet разъемов, коммутируемых с помощью мульти портовых маршрутизаторов. К этим портам можно подключать к видеокамеры для осуществления алгоритмов машинного зрения. Управление блоком также может осуществляться через Ethernet.

В целом, блок управления для автоматизации трактора представляет собой мощное и гибкое устройство, способное управлять различными аспектами работы трактора и подключать различные периферийные устройства. Однако, для использования всех его возможностей, потребуется дополнительное программное и аппаратное обеспечение и настройка.

3. Обобщение подходов к автоматизации БСТС

Рассмотрение автоматизации тракторной техники в рамках данного исследования позволяет выделить ключевые подходы и принципы, которые могут быть обобщены и применены к различным типам сельскохозяйственных транспортных систем (БСТС). В данном разделе представлен обзор основных принципов, использованных при автоматизации тракторов МТЗ-112Н-01 и МТЗ-3522.

Иерархическая система управления

Применение иерархической системы управления является ключевым аспектом при автоматизации тракторов. Разделение системы управления на несколько уровней позволяет эффективно интегрировать различные функции, обеспечивая гибкость и расширяемость.

Классификация подсистем управления

Классификация подсистем управления включает в себя:

- Подсистема реконструкции окружающей обстановки, включая сенсорику и систему машинного зрения.
- Подсистема позиционирования и навигации, формирующая навигационно-временное поле.
- Подсистема коммуникации и связи.
- Вычислительный контур.

Этот подход позволяет эффективно управлять различными аспектами трактора и обеспечивать их взаимодействие.

Принципы бережливого производства и теории ограничений

Принципы бережливого производства и теория ограничений, описанные в предыдущих главах, активно применялись при разработке систем автоматизации БСТС. Эти подходы обеспечивают эффективное управление ограничениями и ресурсами, что является критически важным в условиях сельского хозяйства.

Ретрофит решения

Применение ретрофит решений, таких как автоматизация систем управления серийных тракторов, способствует более быстрой и экономичной интеграции беспилотных технологий в сельском хозяйстве. Этот подход соответствует принципам бережливого производства и поддерживает принцип "бережливого производства".

Применение нейросетевых технологий машинного зрения

В рамках автоматизации сельскохозяйственной техники, в частности тракторов, значительное внимание уделяется применению современных шаблонов проектирования. Одним из таких шаблонов, используемых для повышения эффективности и точности систем автоматизации, является применение нейросетевой технологии машинного зрения YOLO (You Only Look Once)[86].

При этом выбрана самая последняя версия этой технологии – YOLOv8[87] в модификации Tiny со следующей структурной схемой (Рисунок 3.21):

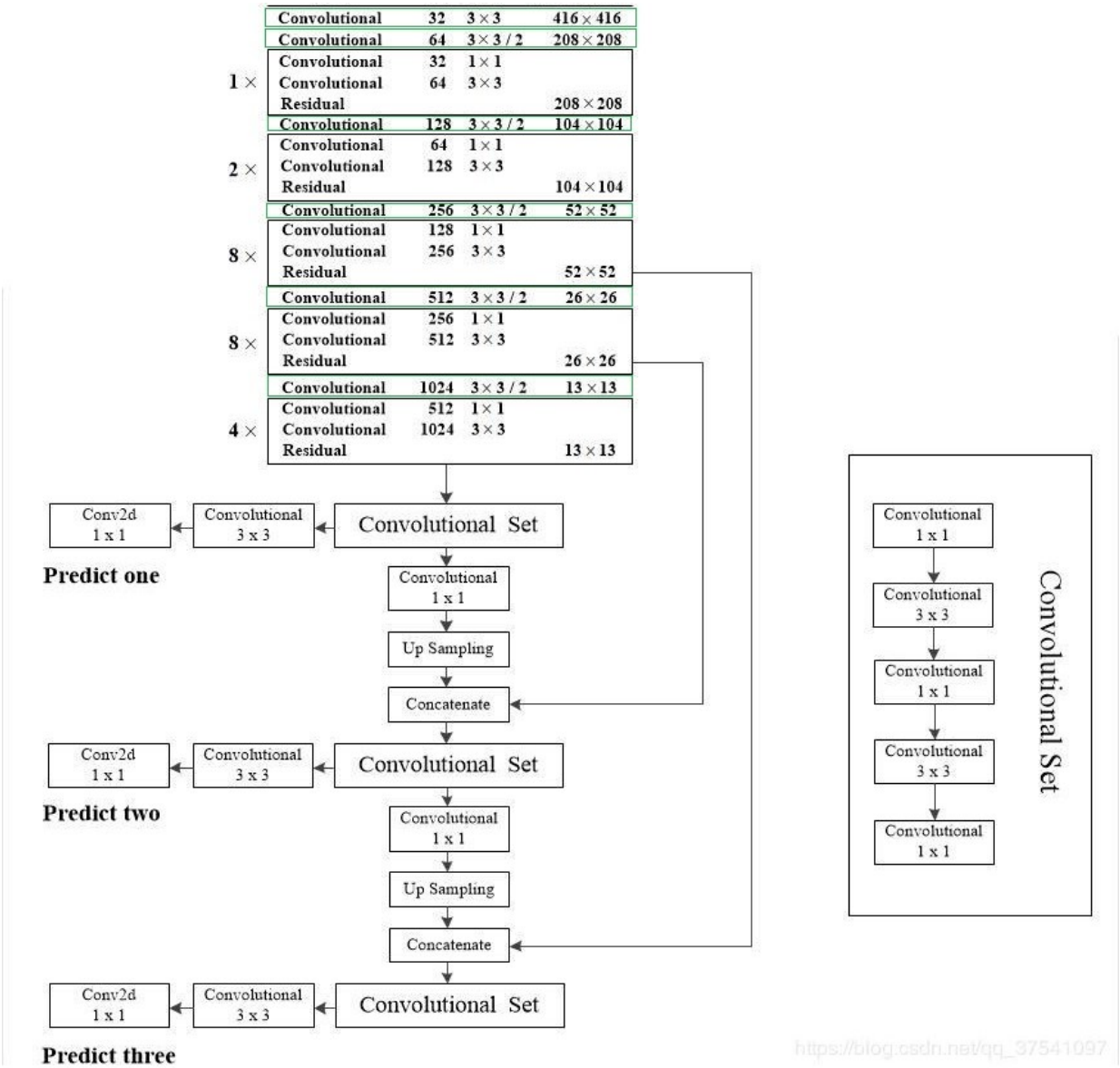


Рисунок 3.21 – Структура нейронной сети YOLO.

На вход этой свёрточной нейронной сети подается двумерное изображение, которое затем обрабатывается свёрточными слоями, причем каждый слой выполняет операции свертки, нормализации (пакетной) и активации. Математическая модель выглядит следующим образом:

В простейшем случае при свертке выходное значение слоя с входным размером (N, C_{in}, H, W) и выходным $(N, C_{out}, H_{out}, W_{out})$ можно записать как

$$out\left(N_i C_{out_j}\right) = bias\left(C_{out_j}\right) + \sum_{k=0}^{C_{in}-1} weight(C_{out_j}, k) * input(N_i, k) \qquad (3.1),$$

где N – размер пакета, C – количество каналов, H высота, W ширина в пикселях, *bias* – тензор компенсации (обучаемый параметр);

Пакетная нормализация –

$$y_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} * \gamma + \beta \qquad (3.2),$$

где μ_B – математическое ожидание пакета, σ_B^2 – дисперсия пакета, γ, β – настраиваемые параметры сжатия и сдвига, ϵ – константа для вычислительной устойчивости.

В качестве функции активации используется *ReLU(Leaky rectified linear unit)* для устранения исчезающего градиента:

$$f(x) = \begin{cases} 0{,}01x, & x < 0 \\ x, & x \geq 0 \end{cases} \qquad (3.3)$$

Все эти аспекты представляет собой синтез основных концепций и практических решений, предложенных в предыдущих разделах, что делает его ключевым компонентом для понимания применяемых подходов к автоматизации сельскохозяйственной техники.

Заключение главы 3

Третья глава исследования представляет собой ключевой этап, где детально рассматривается процесс автоматизации двух моделей тракторов "МТЗ-112Н-01" и "МТЗ-3522". Каждый раздел этой главы представляет собой глубокий анализ технических аспектов, используемых систем и перспектив дальнейшей автоматизации.

В разделе "Автоматизация трактора «МТЗ-112Н-01»" детально описаны технические характеристики трактора, перечислены и описаны автоматизируемые системы, такие как системы управления движением и системы машинного зрения. Подраздел, посвященный используемому программному обеспечению, предоставляет подробный обзор и полный список инструментов, использованных для реализации автоматизации, включая программы управления, алгоритмы машинного зрения и инструменты для обработки данных лидара. Эти приложения обеспечивают не только основные функции управления, но и инновационные возможности, такие как распознавание объектов и автопилотирование.

"Автоматизация трактора «МТЗ-3522»" представляет собой аналогичный подход с подробным описанием технических характеристик, перечня автоматизируемых систем, принципов автоматизации, приводятся сведения об управлении и рассматриваются перспективы для дальнейшего совершенствования.

В разделе "Обобщение подходов к автоматизации БСТС" проанализированы общие принципы, используемые при разработке и внедрении систем автоматизации для беспилотных сельскохозяйственных транспортных систем. Эти принципы и шаблоны проектирования оказываются универсальными в различных сценариях. Этот раздел выступает в роли обобщающего заключения, в котором подчеркиваются общие тренды и применяемые подходы.

Таким образом, третья глава представляет собой комплексное исследование процессов автоматизации, обогащенное конкретными

техническими деталями и перспективами развития. Представленные результаты служат важным вкладом в область беспилотных сельскохозяйственных транспортных систем и могут служить основой для дальнейших исследований и разработок.

Кроме того, эта глава служит переходом к следующему ключевому этапу исследования - виртуальному моделированию. В последующей главе будет рассмотрено, как симуляция может дополнить и улучшить автоматизированные системы, обеспечивая более точное предвидение и оптимизацию работы. Этот этап виртуального моделирования призван не только дополнить текущие решения, но и улучшить процесс проектирования, предоставляя более точные данные для предвидения работы системы и ее оптимизации.

Глава 4. Виртуальное моделирование.

1. Постановка задачи виртуального моделирования

При построении реальных систем беспилотного транспорта следует учитывать, какое оборудование (в том числе и сенсорика) будет установлено в технику. Более того, алгоритмы автопилотирования транспортного средства[88], а также системы распознавания объектов требуют тщательной проверки, так как могут возникнуть ошибки в ходе эксплуатации системы, что может привести к негативным последствиям, в том числе и связанных с риском для жизни людей, находящихся как в транспортных средствах, так и являющихся пешеходами. Такие огромные риски делают задачу виртуального моделирования разрабатываемой системы для беспилотного трактора тяжелого класса актуальной и одной из самых важных[89].

Сегодня распространено большое количество программного обеспечения, которое позволяет выполнять имитационное моделирование различных систем с учетом физики реального мира. В работах [90-92] приведены сравнительные характеристики имеющихся систем виртуального моделирования. Таким образом, было принято решение использовать Gazebo для выполнения виртуального моделирования. Для управления трактором тяжелого класса, так же взаимодействия с его сенсорикой было принято решение использовать экосистему программирования роботов Robot Operating System 2 (ROS2)[93 - 95], которая обладает полной поддержкой Gazebo[96]. В качестве операционной системы (ОС) для разворачивания системы виртуального моделирования было принято решение использовать Ubuntu 22.04.

2. Создание виртуальных моделей

Одним из частей симуляции в Gazebo и ROS2 является виртуальная модель транспортного средства и окружающей обстановки (мира). Для описания виртуальных моделей используются файлы из следующих форматов:

- SDF
- URDF

Оба формата является способом представления описания виртуальных моделей на базе формата XML. Так же файлы описания SDF и URDF позволяют указывать ссылки на визуальные компоненты модели и плагины для элементов виртуальной модели ROS2. Для описания виртуальной модели трактора тяжелого класса был выбран способ описания с помощью формата SDF ввиду своей простоты и наличия полной документацией с подробным описанием [97].

2.1 Создание виртуальной окружающей обстановки

В первую очередь для виртуального моделирования необходимо описывать окружающую обстановку (мира)[98, 99]. Файл описания мира должен состоять из нескольких компонентов: освещение (light), земная поверхность (model), на которой будут располагаться модели, описание сцены (scene), физика мира (physics), состояние мира (state) и начальное расположение камеры наблюдателя (camera)[100].

Для создания файла мира трактора тяжелого класса были описаны следующие компоненты[101]:

- Освещение мира (light):
    - Тип источника света (type) – “directional” (направленный);
    - Тени от источника света (cast_shadows) – включены (1);
    - Позиция (pose) – X = 0 м, Y = 0 м, Z = 10 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 3.14 рад;
    - Рассеяние света (diffuse) – R = 0.8, G = 0.8, B = 0.8, A = 1;
    - Отражающая способность цветов (specular) – R = 0.1, G = 0.1, B = 0.1, A = 1;
    - Затухание света (attenuation):
        - Дальность (range) – 1000 м;
        - Постоянный коэффициент затухания (constant) – 0.9;

        - Линейный коэффициент затухания (linear) – 0.01;
        - Квадратичный коэффициент затухания (quadratic) – 0.001.
    - Направление света (direction) – угол крена = -0.5 рад, угол тангажа = 0.5 рад, угол рыскания = -1 рад.
- Модель поверхности на котором будут размещаться модели (model):
    - Статичность модели (static) – статичная (1);
    - Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного плоскости (plane) размером (size) 1000м x 1000м. Нормаль плоскости (normal) направлена вдоль оси Z (X = 0, Y = 0, Z = 1):
        - Коэффициент трения в направлении первой пирамиды трения (mu) – 100;
        - Коэффициент трения в направлении второй пирамиды трения (mu2) – 50;
        - Отскакивание от поверхности (bounce) – включено
        - Параметры контакта (contact) – использование параметров ODE (ode).
    - Визуальная модель (visual) – геометрическая фигура (geometry) в виде прямоугольного плоскости (plane) размером (size) 1000м x 1000м. Нормаль плоскости (normal) направлена вдоль оси Z (X = 0, Y = 0, Z = 1);
    - Действие гравитации (gravity) – включено (1).
- Сцена окружающей обстановки (scene):
    - Освещение окружения (ambient) – R = 0.2, G = 0.4, B = 0.4, A = 1;
    - Цвета заднего фона (background) – R = 0.7, G = 0.7, B = 0.7, A = 1;
- Физика мира (physics):
    - Тип физики – “ode”;
    - Максимальный размер шага (max_step_size) – 0.01;
    - Коэффициент реального времени (real_time_factor) – 1;
    - Частота обновления реального времени (real_time_update_rate) – 0;

- Гравитация (gravity) – X = 0 м/с², Y = 0 м/с², Z = -9.8 м/с².

- Состояние мира (state):
  - Реальное калибровочное время (real_time) – 0с 44986 нс;
  - Калибровочное время (wall_time) – 1377677575с 940727583 нс;

2.2 Создание виртуального транспортного средства

Виртуальный трактор тяжелого класса состоит из 3 основных частей:

- Описание модели в формате SDF;
- 3D модели частей трактора;
- Материалы и текстуры.

Гибкость Gazebo позволяет использовать 3D модели, в которых текстурирование описано внутри файлов 3D модели[102]. Для трактора тяжелого класса были спроектированы 3D модели колес и основной части трактора используя ПО 3D моделирования SketchUp имеющую бесплатный вариант лицензии [103, 104].

Структура модели виртуального трактора тяжелого класса изображена Рисунке 4.1.

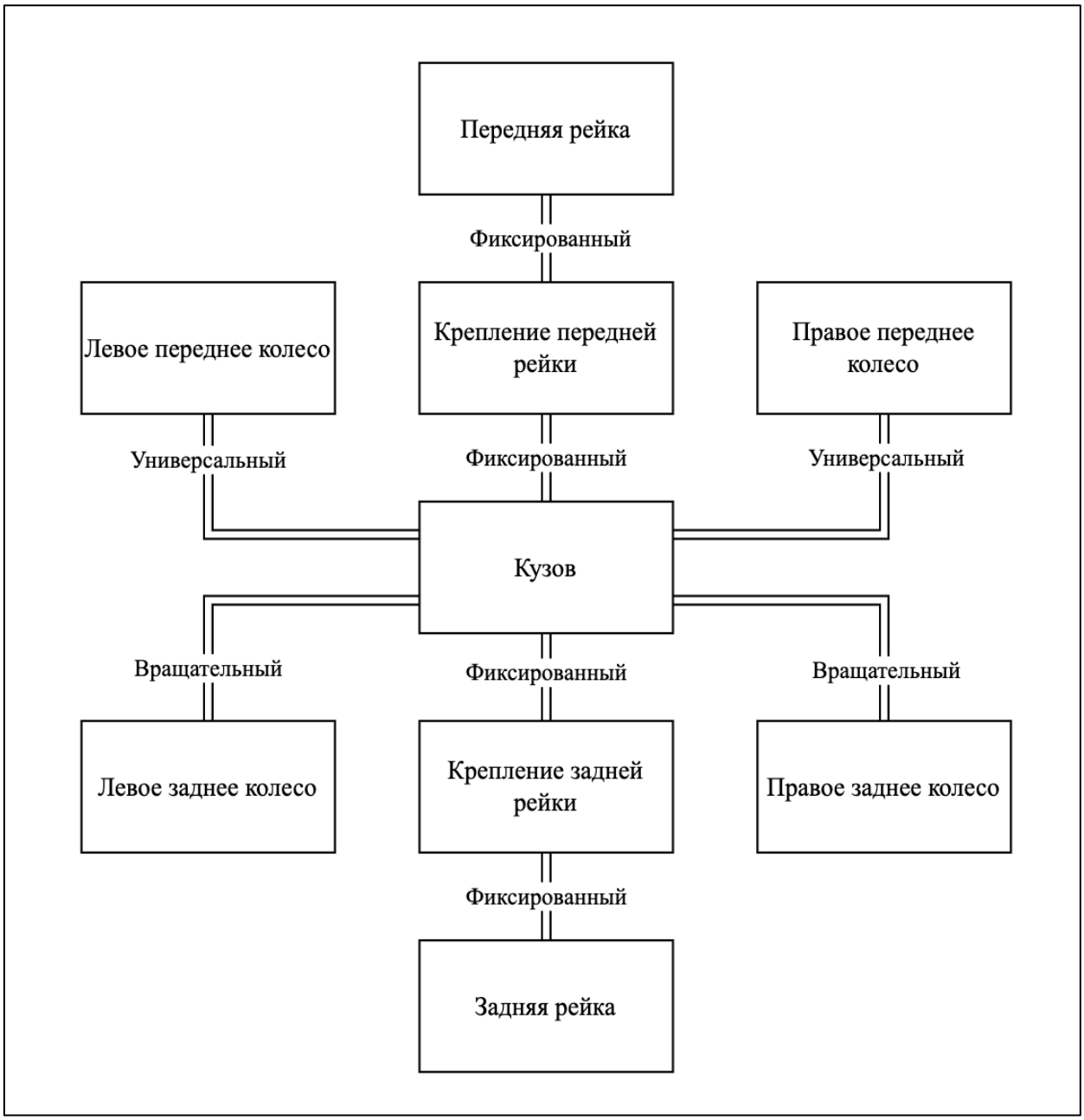


Рисунок 4.1 – Структура виртуального трактора тяжелого класса с указанием типа соединение между звеньями.

SDF описание модели выглядит следующим образом (соединения звеньев указаны на рисунке 4.1):

- Описание звеньев:
  - Звено кузова:
    - Масса звена (mass) – “11700.0” килограмм;
    - Позиция звена (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;

- Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 5.5м x 2.85м x 3.35м, с параметрами инерции (itertia) – ixx = 18861.38, ixy = 0, iyy = 40435.69, ixz = 0, izz = 37413.19;
- Визуальная модель (visual) – геометрическая фигура (geometry) в виде 3d модели каркаса (mesh) с указанием пути, с коэффициентами масштабирования модели (scale) X = 1, Y = 1, Z = 1. Позиция визуальной модели (pose) – X = -2.81 м, Y = 1.425 м, Z = -1.3 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = -1.570796 рад.

- Звено переднего левого колеса:
  - Позиция звена (pose) – X = 1.2805 м, Y = 1.375 м, Z = -0.519615 м, угол крена = 0 рад, угол тангажа = 1.5707 рад, угол рыскания = 1.5707 рад;
  - Масса звена (mass) – “262” килограмм;
  - Модель коллизии (collision) – геометрическая фигура (geometry) в виде цилиндра (cylinder) с радиусом 0.822 м, длиной 0.6м, с параметрами инерции (itertia) – ixx = 0, ixy = 0, iyy = 88.514604, ixz = 0, izz = 0;
    - Коэффициент трения в направлении первой пирамиды трения (mu) – 1;
    - Коэффициент трения в направлении второй пирамиды трения (mu2) – 1;
    - Коэффициент жесткости (kp) – 1e9;
    - Минимально допустимая глубина до применения импульса коррекции контакта (min_depth) – 0.001.
  - Визуальная модель (visual) – геометрическая фигура (geometry) в виде 3d модели колеса, с коэффициентами масштабирования модели (scale) X = 2.5, Y = 2.5, Z = 2.5. Позиция визуальной

модели (pose) – X = -0.96 м, Y = -0.95 м, Z = 0.4 м, угол крена = -1.5707 рад, угол тангажа = 0 рад, угол рыскания = 0 рад.

- Звено переднего правого колеса:
    - Позиция звена (pose) – X = 1.2805 м, Y = -1.375 м, Z = -0.519615 м, угол крена = 0 рад, угол тангажа = 1.5707 рад, угол рыскания = 1.5707 рад;
    - Масса звена (mass) – “262” килограмм;
    - Модель коллизии (collision) – геометрическая фигура (geometry) в виде цилиндра (cylinder) с радиусом 0.822 м, длиной 0.6м, с параметрами инерции (itertia) – ixx = 0, ixy = 0, iyy = 88.514604, ixz = 0, izz = 0;
        - Коэффициент трения в направлении первой пирамиды трения (mu) – 1;
        - Коэффициент трения в направлении второй пирамиды трения (mu2) – 1;
        - Коэффициент жесткости (kp) – 1e9;
        - Минимально допустимая глубина до применения импульса коррекции контакта (min_depth) – 0.001.
    - Визуальная модель (visual) – геометрическая фигура (geometry) в виде 3d модели колеса, с коэффициентами масштабирования модели (scale) X = 2.5, Y = 2.5, Z = 2.5. Позиция визуальной модели (pose) – X = -0.96 м, Y = 0.95 м, Z = -0.4 м, угол крена = 1.5707 рад, угол тангажа = 0 рад, угол рыскания = 0 рад.

- Звено заднего левого колеса:
    - Позиция звена (pose) – X = -1.7195 м, Y = 1.375 м, Z = -0.311115 м, угол крена = 0 рад, угол тангажа = 1.5707 рад, угол рыскания = 1.5707 рад;
    - Масса звена (mass) – “525” килограмм;

- Модель коллизии (collision) – геометрическая фигура (geometry) в виде цилиндра (cylinder) с радиусом 1.0305 м, длиной 0.71м, с параметрами инерции (itertia) – ixx = 0, ixy = 0, iyy = 278.75669062, ixz = 0, izz = 0;
    - Коэффициент трения в направлении первой пирамиды трения (mu) – 1.1;
    - Коэффициент трения в направлении второй пирамиды трения (mu2) – 1.1;
    - Коэффициент жесткости (kp) – 1e9;
    - Минимально допустимая глубина до применения импульса коррекции контакта (min_depth) – 0.001.
- Визуальная модель (visual) – геометрическая фигура (geometry) в виде 3d модели колеса, с коэффициентами масштабирования модели (scale) X = 2.8, Y = 2.8, Z = 2.8. Позиция визуальной модели (pose) – X = -1.07 м, Y = -1.07 м, Z = 0.4 м, угол крена = -1.5707 рад, угол тангажа = 0 рад, угол рыскания = 0 рад.

- Звено заднего правого колеса:
    - Позиция звена (pose) – X = -1.7195 м, Y = -1.375 м, Z = -0.311115 м, угол крена = 0 рад, угол тангажа = 1.5707 рад, угол рыскания = 1.5707 рад;
    - Масса звена (mass) – “525” килограмм;
    - Модель коллизии (collision) – геометрическая фигура (geometry) в виде цилиндра (cylinder) с радиусом 1.0305 м, длиной 0.71м, с параметрами инерции (itertia) – ixx = 0, ixy = 0, iyy = 278.75669062, ixz = 0, izz = 0;
        - Коэффициент трения в направлении первой пирамиды трения (mu) – 1.1;
        - Коэффициент трения в направлении второй пирамиды трения (mu2) – 1.1;

        - Коэффициент жесткости (kp) – 1e9;
        - Минимально допустимая глубина до применения импульса коррекции контакта (min_depth) – 0.001.
    - Визуальная модель (visual) – геометрическая фигура (geometry) в виде 3d модели колеса, с коэффициентами масштабирования модели (scale) X = 2.8, Y = 2.8, Z = 2.8. Позиция визуальной модели (pose) – X = -1.07 м, Y = 1.07 м, Z = -0.4 м, угол крена = 1.5707 рад, угол тангажа = 0 рад, угол рыскания = 0 рад.
- Звено крепления передней рейки:
    - Позиция звена (pose) – X = 2.4 м, Y = 0 м, Z = -0.6 м, угол крена = 0 рад, угол тангажа = 0.785 рад, угол рыскания = 0 рад;
    - Масса звена (mass) – “15.0” килограмм;
    - Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) с параметрами: длина 1 м, ширина 0.5 м, высота 0.041;
    - Визуальная модель (visual) совпадает с моделью коллизии.
- Звено крепления задней рейки:
    - Позиция звена (pose) – X = -2.5 м, Y = 0 м, Z = -0.6 м, угол крена = 0 рад, угол тангажа = -0.785 рад, угол рыскания = 0 рад;
    - Масса звена (mass) – “15.0” килограмм;
    - Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) с параметрами: длина 1 м, ширина 0.5 м, высота 0.041;
    - Визуальная модель (visual) совпадает с моделью коллизии.
- Звено передней рейки:
    - Позиция звена (pose) – X = 2.8 м, Y = 0 м, Z = -0.94 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
    - Масса звена (mass) – “30.0” килограмм;

- Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) с параметрами: длина 0.05 м, ширина 3.05 м, высота 0.041;
- Визуальная модель (visual) совпадает с моделью коллизии.

- Звено задней рейки:
  - Позиция звена (pose) – X = -2.9 м, Y = 0 м, Z = -0.94 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
  - Масса звена (mass) – “30.0” килограмм;
  - Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) с параметрами: длина 0.05 м, ширина 3.05 м, высота 0.041;
  - Визуальная модель (visual) совпадает с моделью коллизии.

- Описание соединений:
  - Соединение каркаса с передним левым колесом:
    - Тип соединения (joint type) – универсальный (“universal”);
    - Родительское звено (parent) – кузов;
    - Дочернее звено (child) – переднее левое колесо;
    - Оси поворота (axis) – Z;
    - Ограничение угла нижней границы поворота – -0.8727 рад;
    - Ограничение угла высшей границы поворота – 0.8727 рад;
    - Оси вращения (axis2) – Y;
    - Значение физического статического трения – 18.0474092253;
    - Позиция соединения – X = 0 м, Y = 0 м, Z = 0 м, угол крена = -0.08726646259971647 рад, угол тангажа = 0 рад, угол рыскания = 0 рад.
  - Соединение кузова с передним правым колесом:
    - Тип соединения (joint type) – универсальный (“universal”);
    - Родительское звено (parent) – кузов;
    - Дочернее звено (child) – переднее правое колесо;
    - Ось поворота (axis) – Z;

- Ограничение угла нижней границы поворота – -0.8727 рад;
- Ограничение угла высшей границы поворота – 0.8727 рад;
- Ось вращения (axis2) – Y;
- Значение физического статического трения – 18.0474092253;
- Позиция соединения – X = 0 м, Y = 0 м, Z = 0 м, угол крена = -0.08726646259971647 рад, угол тангажа = 0 рад, угол рыскания = 0 рад.

- Соединение кузова с задним левым колесом:
  - Тип соединения (joint type) – вращательный (“revolute”);
  - Родительское звено (parent) – кузов;
  - Дочернее звено (child) – заднее левое колесо;
  - Оси вращения (axis) – Y;
  - Значение физического статического трения – 12.0031606150200002;
- Соединение кузова с задним правым колесом:
  - Тип соединения (joint type) – вращательный (“revolute”);
  - Родительское звено (parent) – кузов;
  - Дочернее звено (child) – заднее правое колесо;
  - Оси вращения (axis) – Y;
  - Значение физического статического трения – 12.0031606150200002;
- Описание плагинов:
  - Плагин привода по Аккерману [105-107], необходимый для движения автомобиля:
    - Файл плагина (filename) – “libgazebo_ros_ackermann_drive.so”;
    - Частота обновления – 100 Гц;
    - Максимальный абсолютный угол поворота колес – 0.6981;
    - Максимальный абсолютный угол поворота руля – 7.85;
    - Максимальная абсолютная линейная скорость – 2.2 м/с;

- Коэффициенты левого ПИД-регулятора: пропорциональный – 20000, интегральный – 0, дифференциальный – 1;
- Коэффициенты правого ПИД-регулятора: пропорциональный – 20000, интегральный – 0, дифференциальный – 1;
- Коэффициенты ПИД-регулятора линейной скорости: пропорциональный – 1000, интегральный – 0, дифференциальный – 1;
- Базовый элемент транспортного средства – кузов;
- Узел чтения показаний одометра – “odom_demo”.

- Плагин публикации состояния соединений:
  - Пространство имен для формирования выходных данных (robotNamespace) – “/tracktor”;
  - Соединения для чтения данных: соединение кузова с левым передним колесом, соединение кузова с правым передним колесом.

2.3 Создание системы сенсорики (виртуальные датчики)

В инфраструктуре Gazebo и ROS2 сенсорика виртуального трактора тяжелого класса может являться частью модели виртуального транспортного средства[108]. Для обеспечения полной работоспособности системы[109-111] беспилотного трактора тяжелого класса было принято решение использовать следующую сенсорику:

- 8 сонаров;
- Лидар;
- Камера;
- 2 GPS датчика.

Дополненная структурная схема модели трактора с установленной сенсорикой показана на Рисунке 4.2.

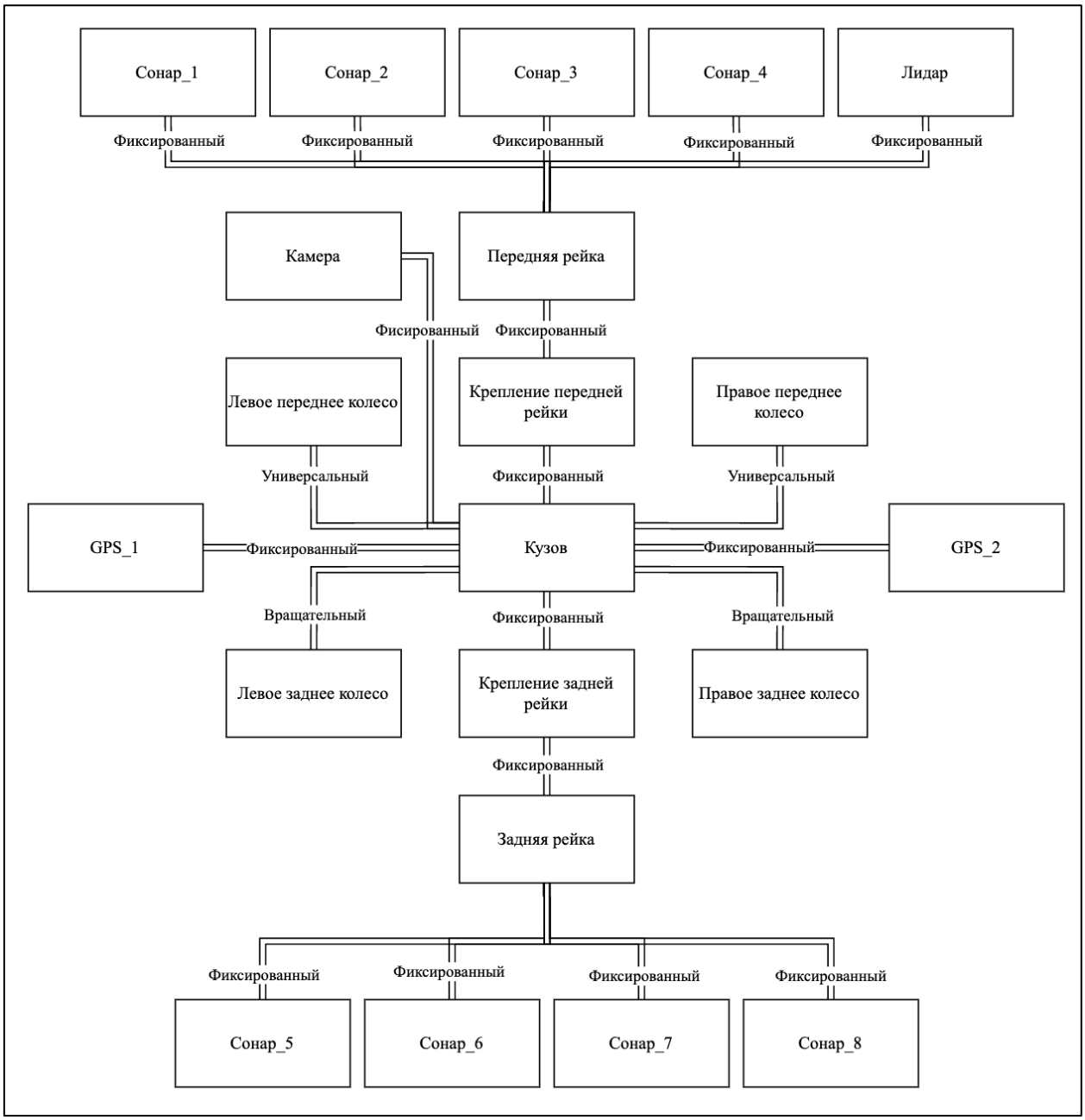


Рисунок 4.2 – Структурная схема виртуального трактора тяжелого класса с установленной сенсорикой с указанием типа соединений между звеньями.

SDF описание сенсорики модели выглядит следующим образом (соединения звеньев указаны на Рисунке 4.2):

- Описание звеньев:
  - Звено лидара:
    - Масса звена (mass) – 0.1 кг;

- Позиция звена (pose) – X = 2.8 м, Y = 0 м, Z = -0.8 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
- Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = -0.0145 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
- Визуальная модель (visual) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.2м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = -0.0145 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
- Сенсор (sensor) типа луч (ray):
  - Позиция сенсора (pose) – X = 0.01 м, Y = 0 м, Z = 0.0175 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
  - Количество горизонтальных лучей (samples) – 8;
  - Горизонтальное разрешение (resolution) – 1;
  - Минимальный горизонтальный угол (min_angle) – -0.34 рад;
  - Максимальный горизонтальный угол (max_angle)– 0.34 рад;
  - Дальность (range) минимальная (min) – 0.01 м;
  - Дальность (range) максимальная (max) – 20 м;
  - Плагин libgazebo_ros_ray_sensor.so:
    - Узел для чтения данных – “~/out:=lidar”;
    - Тип выходных данных (output_type)– “sensor_msgs/LaserScan”;
    - Идентификатор (frame_name) – “lidar_link”.

- Звено камеры:

- Масса звена (mass) – 0.1 кг;
- Позиция звена (pose) – X = 2.6 м, Y = 0 м, Z = -0.3 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
- Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.1м x 0.1м x 0.1м;
- Визуальная модель (visual) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.1м x 0.1м x 0.1м;
- Сенсор (sensor) типа луч (camera):
  - Горизонтальное поле зрения (horizontal_fov) – 1.9198 рад;
  - Ширина (width) – 1280 пикселей;
  - Высота (height) – 720 пикселей;
  - Минимальная дальность (near) – 0.1 м;
  - Максимальная дальность (far) – 100 м;
  - Плагин libgazebo_ros_camera.so:
    - Узел для чтения информации камеры (cameraInfoTopicName) – “camera_info”;
    - Узел чтения кадров камеры () – “image_raw”;
    - Идентификатор (frameName) – “camera_link_optical”.

- Звено GPS 1:
  - Масса звена (mass) – 0.1 кг;
  - Позиция звена (pose) – X = 2.5 м, Y = 0 м, Z = 0.55 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
  - Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.1м x 0.1м x 0.1м;

- Визуальная модель (visual) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.1м x 0.1м x 0.1м;
- Сенсор (sensor) типа GPS:
  - Частота обновления 20 Гц;
  - Плагин libgazebo_ros_gps_sensor.so:
    - Узел чтения данных GPS – “gps_1_controller/out:=gps1”;
    - Идентификатор (frameName) – “gps_1_link”.

- Звено GPS 2:
  - Масса звена (mass) – 0.1 кг;
  - Позиция звена (pose) – X = -2.0 м, Y = 0 м, Z = 0.55 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
  - Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.1м x 0.1м x 0.1м;
  - Визуальная модель (visual) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.1м x 0.1м x 0.1м;
  - Сенсор (sensor) типа GPS:
    - Частота обновления 20 Гц;
    - Плагин libgazebo_ros_gps_sensor.so:
      - Узел чтения данных GPS – “gps_2_controller/out:=gps2”;
      - Идентификатор (frameName) – “gps_2_link”.

- Звено сонара 1:
  - Позиция звена (pose) – X = 2.8 м, Y = 0.6875 м, Z = -0.9 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
  - Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size)

0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;

- Визуальная модель (visual) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
- Сенсор (sensor) типа сонар:
  - Позиция сенсора (pose) – X = 0.05 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
  - Количество горизонтальных лучей (samples) – 20;
  - Горизонтальное разрешение (resolution) – 1;
  - Минимальный горизонтальный угол (min_angle) – -0.69 рад;
  - Максимальный горизонтальный угол (max_angle)– 0.69 рад;
  - Количество вертикальных лучей (samples) – 5;
  - Вертикальное разрешение (resolution) – 1;
  - Минимальный вертикальный угол (min_angle) – -0.1 рад;
  - Максимальный вертикальный угол (max_angle)– 0.1 рад;
  - Дальность (range) минимальная (min) – 0 м;
  - Дальность (range) максимальная (max) – 2 м;
  - Плагин libgazebo_ros_ray_sensor.so:
    - Узел для чтения данных – “~/out:=sonar1”;
    - Тип излучения (radiation type) – “ultrasonic”;
    - Тип выходных данных (output_type)– “sensor_msgs/LaserScan”;
    - Идентификатор (frame_name)– “sonar_link_1”.

- Звено сонара 2:

- Позиция звена (pose) – X = 2.8 м, Y = -0.6875 м, Z = -0.9 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
- Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
- Визуальная модель (visual) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
- Сенсор (sensor) типа сонар:
    - Позиция сенсора (pose) – X = 0.05 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
    - Количество горизонтальных лучей (samples) – 20;
    - Горизонтальное разрешение (resolution) – 1;
    - Минимальный горизонтальный угол (min_angle) – -0.69 рад;
    - Максимальный горизонтальный угол (max_angle)– 0.69 рад;
    - Количество вертикальных лучей (samples) – 5;
    - Вертикальное разрешение (resolution) – 1;
    - Минимальный вертикальный угол (min_angle) – -0.1 рад;
    - Максимальный вертикальный угол (max_angle)– 0.1 рад;
    - Дальность (range) минимальная (min) – 0 м;
    - Дальность (range) максимальная (max) – 2 м;
    - Плагин libgazebo_ros_ray_sensor.so:
        - Узел для чтения данных – “~/out:=sonar2”;
        - Тип излучения (radiation type) – “ultrasonic”;

- Тип выходных данных (output_type)– “sensor_msgs/LaserScan”;
- Идентификатор (frame_name)– “sonar_link_2”.

- Звено сонара 3:
  - Позиция звена (pose) – X = 2.8 м, Y = -1.375 м, Z = -0.9 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
  - Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
  - Визуальная модель (visual) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
  - Сенсор (sensor) типа сонар:
    - Позиция сенсора (pose) – X = 0.05 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
    - Количество горизонтальных лучей (samples) – 20;
    - Горизонтальное разрешение (resolution) – 1;
    - Минимальный горизонтальный угол (min_angle) – -0.69 рад;
    - Максимальный горизонтальный угол (max_angle)– 0.69 рад;
    - Количество вертикальных лучей (samples) – 5;
    - Вертикальное разрешение (resolution) – 1;
    - Минимальный вертикальный угол (min_angle) – -0.1 рад;
    - Максимальный вертикальный угол (max_angle)– 0.1 рад;
    - Дальность (range) минимальная (min) – 0 м;

        - Дальность (range) максимальная (max) – 2 м;
        - Плагин libgazebo_ros_ray_sensor.so:
            - Узел для чтения данных – “~/out:=sonar3”;
            - Тип излучения (radiation type) – “ultrasonic”;
            - Тип выходных данных (output_type)– “sensor_msgs/LaserScan”;
            - Идентификатор (frame_name)– “sonar_link_3”.
- Звено сонара 4:
    - Позиция звена (pose) – X = 2.8 м, Y = 1.375 м, Z = -0.9 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
    - Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
    - Визуальная модель (visual) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
    - Сенсор (sensor) типа сонар:
        - Позиция сенсора (pose) – X = 0.05 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
        - Количество горизонтальных лучей (samples) – 20;
        - Горизонтальное разрешение (resolution) – 1;
        - Минимальный горизонтальный угол (min_angle) – -0.69 рад;
        - Максимальный горизонтальный угол (max_angle)– 0.69 рад;
        - Количество вертикальных лучей (samples) – 5;

        - Вертикальное разрешение (resolution) – 1;
        - Минимальный вертикальный угол (min_angle) – -0.1 рад;
        - Максимальный вертикальный угол (max_angle)– 0.1 рад;
        - Дальность (range) минимальная (min) – 0 м;
        - Дальность (range) максимальная (max) – 2 м;
        - Плагин libgazebo_ros_ray_sensor.so:
            - Узел для чтения данных – “~/out:=sonar4”;
            - Тип излучения (radiation type) – “ultrasonic”;
            - Тип выходных данных (output_type)– “sensor_msgs/LaserScan”;
            - Идентификатор (frame_name)– “sonar_link_4”.
- Звено сонара 5:
    - Позиция звена (pose) – X = -2.9 м, Y = 0.6875 м, Z = -0.9 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 3.141593 рад;
    - Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
    - Визуальная модель (visual) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
    - Сенсор (sensor) типа сонар:
        - Позиция сенсора (pose) – X = 0.05 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
        - Количество горизонтальных лучей (samples) – 20;
        - Горизонтальное разрешение (resolution) – 1;

        - Минимальный горизонтальный угол (min_angle) – -0.69 рад;
        - Максимальный горизонтальный угол (max_angle)– 0.69 рад;
        - Количество вертикальных лучей (samples) – 5;
        - Вертикальное разрешение (resolution) – 1;
        - Минимальный вертикальный угол (min_angle) – -0.1 рад;
        - Максимальный вертикальный угол (max_angle)– 0.1 рад;
        - Дальность (range) минимальная (min) – 0 м;
        - Дальность (range) максимальная (max) – 2 м;
        - Плагин libgazebo_ros_ray_sensor.so:
            - Узел для чтения данных – “~/out:=sonar5”;
            - Тип излучения (radiation type) – “ultrasonic”;
            - Тип выходных данных (output_type)– “sensor_msgs/LaserScan”;
            - Идентификатор (frame_name)– “sonar_link_5”.
- Звено сонара 6:
    - Позиция звена (pose) – X = -2.9 м, Y = -0.6875 м, Z = -0.9 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 3.141593 рад;
    - Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
    - Визуальная модель (visual) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;

- Сенсор (sensor) типа сонар:
  - Позиция сенсора (pose) – X = 0.05 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
  - Количество горизонтальных лучей (samples) – 20;
  - Горизонтальное разрешение (resolution) – 1;
  - Минимальный горизонтальный угол (min_angle) – -0.69 рад;
  - Максимальный горизонтальный угол (max_angle)– 0.69 рад;
  - Количество вертикальных лучей (samples) – 5;
  - Вертикальное разрешение (resolution) – 1;
  - Минимальный вертикальный угол (min_angle) – -0.1 рад;
  - Максимальный вертикальный угол (max_angle)– 0.1 рад;
  - Дальность (range) минимальная (min) – 0 м;
  - Дальность (range) максимальная (max) – 2 м;
  - Плагин libgazebo_ros_ray_sensor.so:
    - Узел для чтения данных – “~/out:=sonar6”;
    - Тип излучения (radiation type) – “ultrasonic”;
    - Тип выходных данных (output_type)– “sensor_msgs/LaserScan”;
    - Идентификатор (frame_name)– “sonar_link_6”.

- Звено сонара 7:
  - Позиция звена (pose) – X = -2.9 м, Y = -1.375 м, Z = -0.9 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 3.141593 рад;
  - Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;

- Визуальная модель (visual) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
- Сенсор (sensor) типа сонар:
  - Позиция сенсора (pose) – X = 0.05 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
  - Количество горизонтальных лучей (samples) – 20;
  - Горизонтальное разрешение (resolution) – 1;
  - Минимальный горизонтальный угол (min_angle) – -0.69 рад;
  - Максимальный горизонтальный угол (max_angle)– 0.69 рад;
  - Количество вертикальных лучей (samples) – 5;
  - Вертикальное разрешение (resolution) – 1;
  - Минимальный вертикальный угол (min_angle) – -0.1 рад;
  - Максимальный вертикальный угол (max_angle)– 0.1 рад;
  - Дальность (range) минимальная (min) – 0 м;
  - Дальность (range) максимальная (max) – 2 м;
  - Плагин libgazebo_ros_ray_sensor.so:
    - Узел для чтения данных – “~/out:=sonar7”;
    - Тип излучения (radiation type) – “ultrasonic”;
    - Тип выходных данных (output_type)– “sensor_msgs/LaserScan”;
    - Идентификатор (frame_name)– “sonar_link_7”.

- Звено сонара 8:
  - Позиция звена (pose) – X = -2.9 м, Y = 1.375 м, Z = -0.9 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 3.141593 рад;

- Модель коллизии (collision) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
- Визуальная модель (visual) – геометрическая фигура (geometry) в виде прямоугольного параллелепипеда (box) размером (size) 0.05м x 0.05м x 0.041м. Позиция модели коллизии (pose) – X = 0 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
- Сенсор (sensor) типа сонар:
  - Позиция сенсора (pose) – X = 0.05 м, Y = 0 м, Z = 0 м, угол крена = 0 рад, угол тангажа = 0 рад, угол рыскания = 0 рад;
  - Количество горизонтальных лучей (samples) – 20;
  - Горизонтальное разрешение (resolution) – 1;
  - Минимальный горизонтальный угол (min_angle) – -0.69 рад;
  - Максимальный горизонтальный угол (max_angle)– 0.69 рад;
  - Количество вертикальных лучей (samples) – 5;
  - Вертикальное разрешение (resolution) – 1;
  - Минимальный вертикальный угол (min_angle) – -0.1 рад;
  - Максимальный вертикальный угол (max_angle)– 0.1 рад;
  - Дальность (range) минимальная (min) – 0 м;
  - Дальность (range) максимальная (max) – 2 м;
  - Плагин libgazebo_ros_ray_sensor.so:
    - Узел для чтения данных – “~/out:=sonar8”;
    - Тип излучения (radiation type) – “ultrasonic”;
    - Тип выходных данных (output_type)– “sensor_msgs/LaserScan”;

- Идентификатор (frame_name)– “sonar_link_8”.

Трактор тяжелого класса, полученный в результате собранной модели в Gazebo показан на Рисунке 4.3.

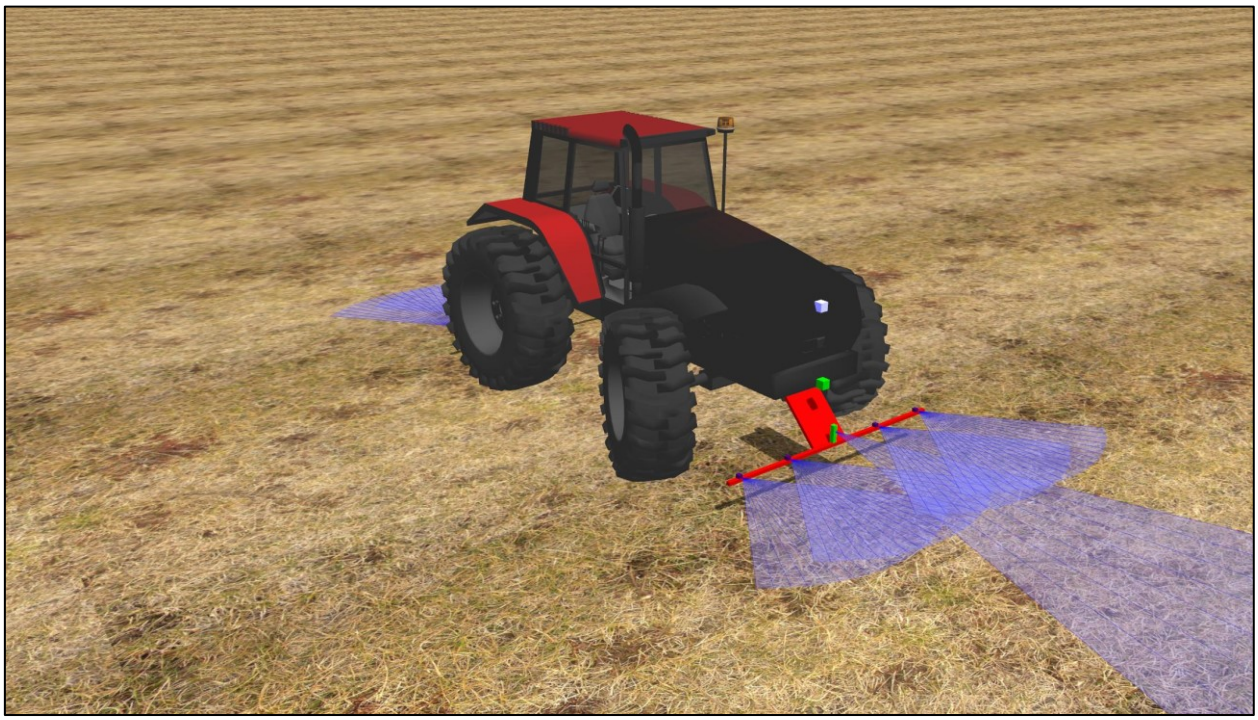


Рисунок 4.3 – Модель трактора тяжелого класса с установленной сенсорикой в Gazebo.

2.4 Разработка алгоритмов управления виртуального БСТС

В инфраструктуре Gezebo и ROS2 алгоритмы управления виртуальным транспортом[112-115] осуществляется с помощью проектных программных пакетов, которые могут осуществлять соединение к проекту симуляции виртуального полигона. Сборка проектного пакета осуществляется с помощью системы сборки colcon [116]. Проектный пакет имеет следующую структуру:

- Директория основного исходного кода программы – содержит в себе основные исходные коды программ, в том числе и алгоритмы управление виртуального БСТС, и узлы симуляции в ROS2 Gazebo;
- Директория запуска программного пакета – содержит в себе файл запуска проекта с указанием узлов симуляции программного пакета;

• Директория ресурсов – содержит в себе дополнительные ресурсы программного пакета;

• Директория тестирования – содержит в себе;

Для разработки алгоритмов управления виртуального БСТС был выбран язык программирования Python 3.10 с использованием компонентов PyQt[117] для обеспечения взаимодействия между частями программы. Для обеспечения работоспособности программы были использовании следующие модули и компоненты:

• os;

• sys;

• traceback;

• rclpy;

• PyQt5;

• gazebo_msgs;

• std_msgs;

• geometry_msgs;

• nav_msgs;

• sensor_msgs;

• threading;

• numpy;

• struct;

• time;

• datetime.

Алгоритм состоит из следующих компонентов:

• Блок управления;

• Блок обратной связи виртуального БСТС;

Блок управления выполняет непосредственное управление трактором тяжелого класса на основе полученных данных по сокету. Сообщения, получаемые сокетом, содержат в себе информацию о состоянии нажатия на

педаль газа, положение трансмиссии, состояние нажатия на тормоз и положение руля. Источником данных может быть любое программное обеспечение, которое передает данные согласно протоколу, указанному в Таблице 4.1. Структурная схема данного блока представлена на Рисунке 4.4.

Таблица 4.1 – Протокол управления виртуальным трактором (Протокол пакета #1).

Номер байта	1	2	3	4	5	6	7
Тип данных	char	char	int8_t	uint8_t	int8_t	char	char
Описание	префикс	номер пакета	газ	тормоз	руль	трансмиссия	суффикс
Данные (диапазон данных)	Символ '#'	Значение 1	Значение от -100 до 100	Значение 0 (отжат) или 1 (нажат)	Значение от -100 до 100	«N» - нейтраль «D» - движение вперед «R» - задний ход	Символ '*'

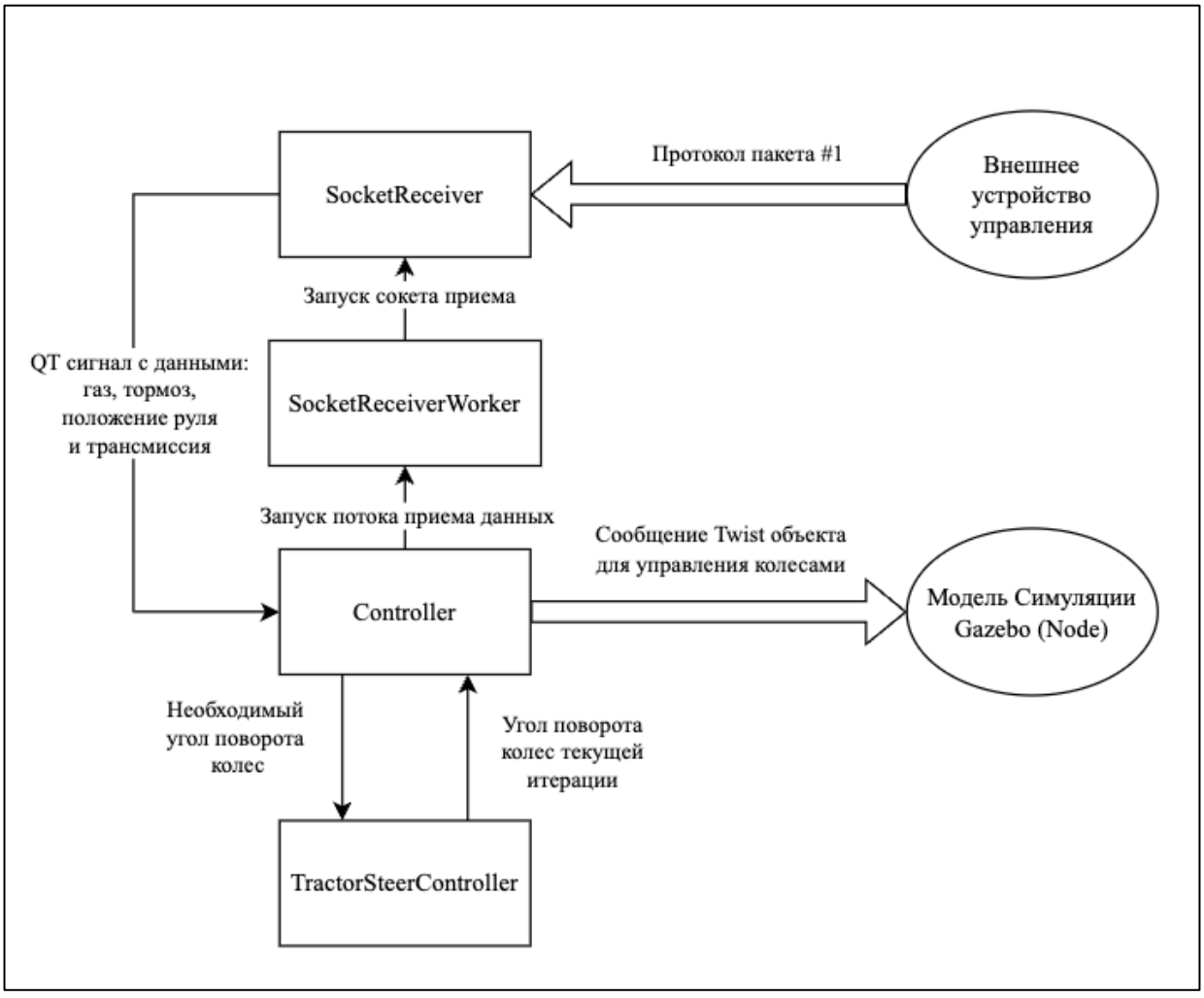


Рисунок 4.4 – Структурная схема блока управления.

Буфер приема сокета объекта класса SocketReceiver при получении данных используется для проверки принадлежности пакета, согласно его структуре, указанной в таблице 4.1, а так же для извлечения данных из полученного пакета и отправки их в контроллер для анализа и формирования управляющего пакета в симуляцию Gazebo.

Класс TractorSteerController выполняет расчет угла, необходимый для поворота колеса с учетом физических характеристик трактора с учетом того, что транспортное средство не может моментально устанавливать необходимый угол поворота колес. Блок-схема алгоритма расчета представлен на Рисунке 4.5.

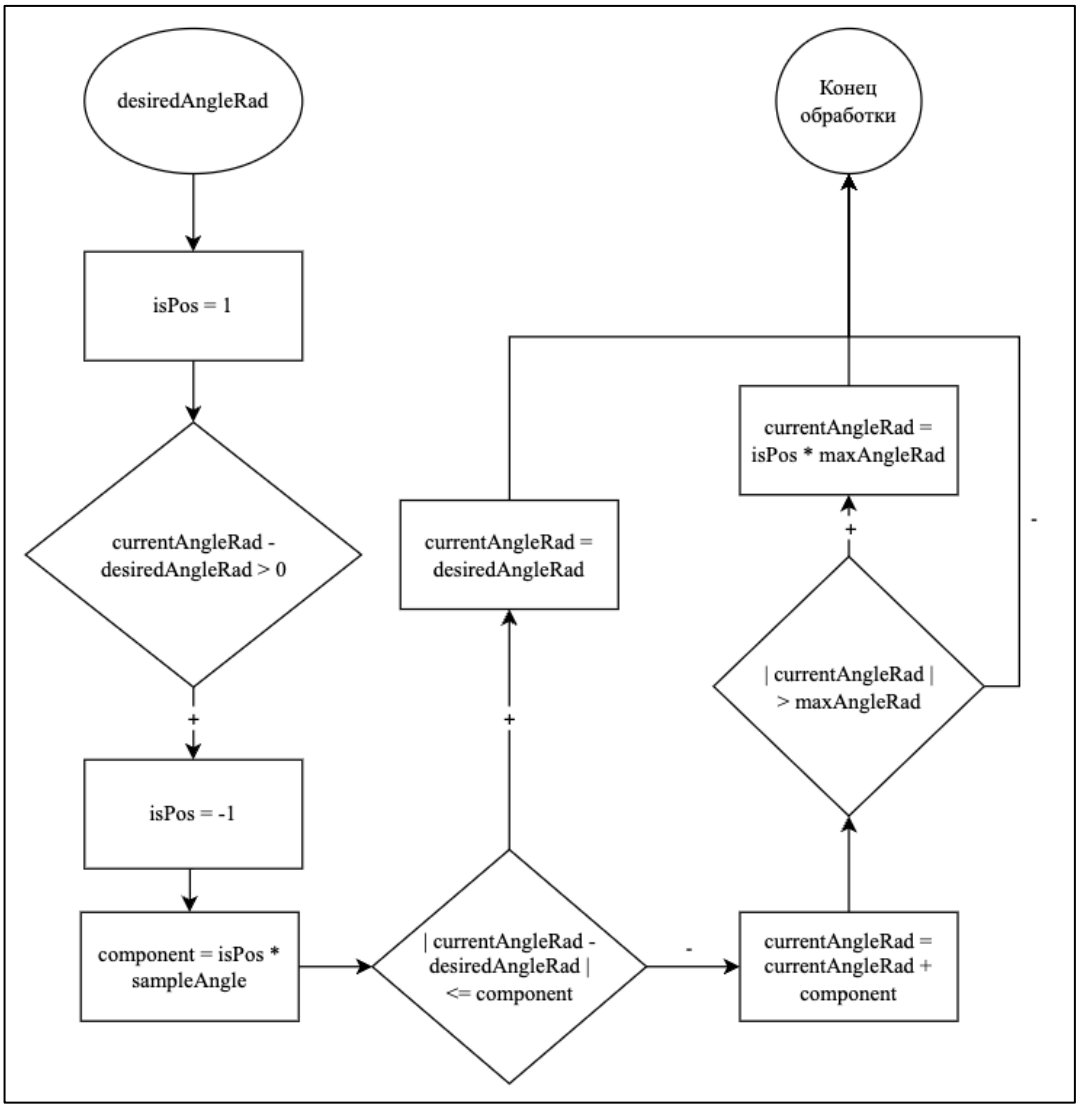


Рисунок 4.5 – Блок-схема алгоритма расчета угла поворота.

Начальное положение угла колес равна нулю (currentAngleRad = 0). Алгоритм на вход получает желаемый угол поворота колес (desiredAngleRad).

Знак направления поворота колеса определяется с помощью переменной isPos. Значение угла поворота сэмпла данных (sampleAngle) определяется по формуле 4.1:

$$sampleAngle = angularSpeedRad * sampleTime \quad (4.1),$$

где:

angularSpeedRad – максимальная угловая скорость колеса, определяемая по формуле 4.2,

sampleTime – 0.2 секунды.

Максимальная угловая скорость поворота колеса (angularSpeedRad) определяется по формуле 4.2:

$$angularSpeedRad = \frac{maxAngleRad}{steerRotationSec} \quad (4.2),$$

где:

maxAngleRad – максимальный угол поворота колеса радианах, соответствующий углу 40 градусов,

steerRotationSec – время поворота колеса до максимального значения угла, равный 2.5 с.

Объект класса Controller обеспечивает взаимодействие между всеми компонентами блока управления. Он выполняет запуск потока приема данных, который выполняет в отдельном потоке чтение данных по сокету в блокирующем режиме, а также выполняет анализ и формирование управляющего пакета, предназначенного для модели симуляции виртуального трактора тяжелого класса. Блок-схема алгоритма анализа данных управления показан на Рисунке 4.6. На вход алгоритма поступают полученные значения газа (gas), поворота руля (wheel) и индикатор нажатия на тормоз (brk), а также показания проекций скоростей, полученных по оси X (speedX) и по оси Y (speedY), получаемые от блока обратной связи виртуальной БСТС.

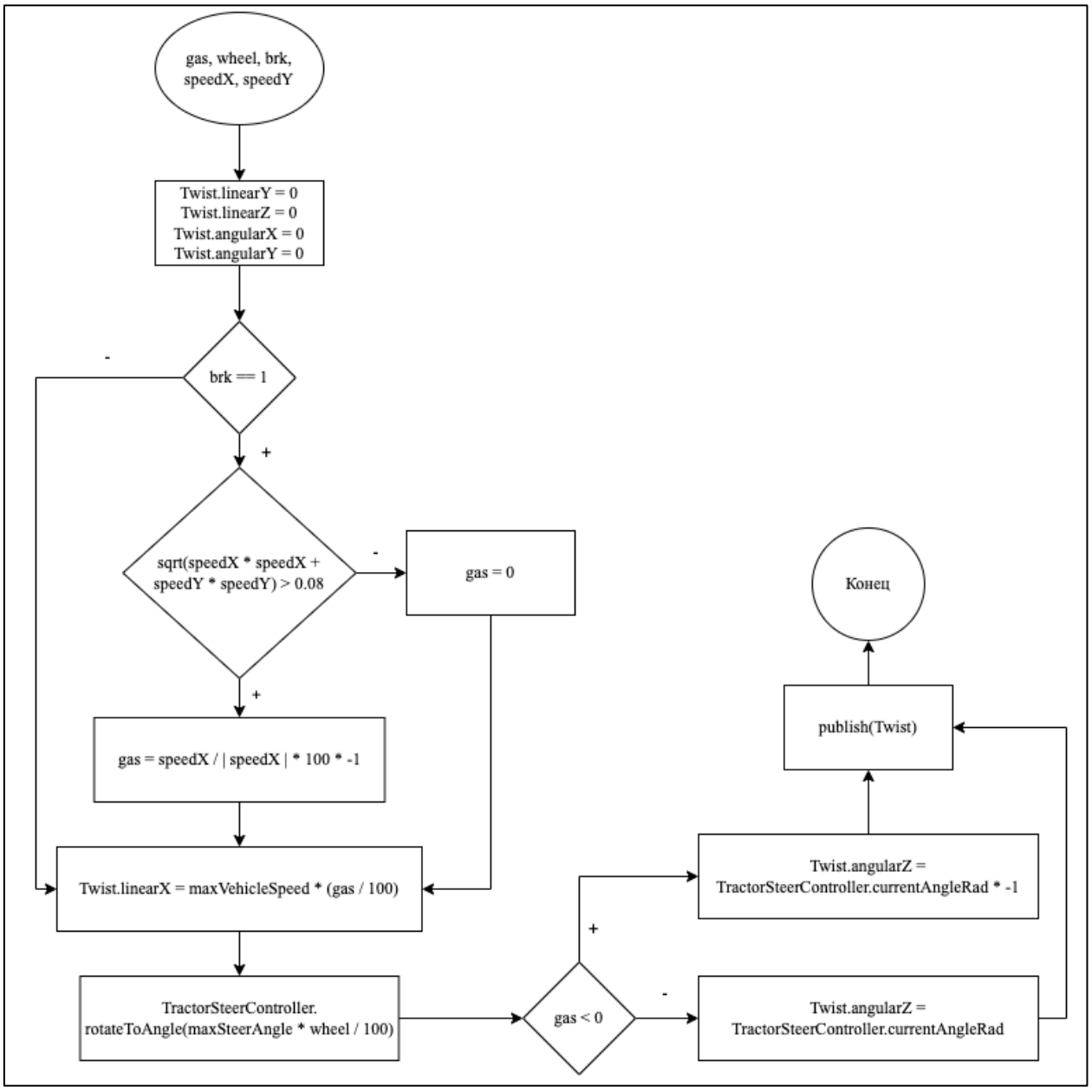


Рисунок 4.6 – Блок-схема алгоритма анализа данных управления.

Максимальная скорость виртуального транспортного средства (maxVehicleSpeed) ограничивается значением в 4 м/с, а угол поворота колес (maxSteerAngle) в 0.6981 рад. Модель виртуального трактора тяжелого класса во время симуляции может принимать управляющие сообщения, в зависимости от использованных компонентов модели. В случае виртуального трактора тяжелого класса используется объект класса Twist для формирования управляющих сообщений. Twist объект в свой структуре состоит из следующих параметров:

- Проекция линейной скорости по оси X;
- Проекция линейной скорости по оси Y;
- Проекция линейной скорости по оси Z;
- Проекция угловой скорости на ось X;
- Проекция угловой скорости на ось Y;
- Проекция угловой скорости на ось Z;

При этом используются система координат относительно модели виртуального трактора. Стоит отметить, что в описанной модели имеется плагин управления транспортного средства с учетом геометрии рулевого управления Аккермана, поэтому показания проекций угловых скоростей несет другое функциональное назначение, а именно – значения углов поворота колес относительно осей. Полученный в результате работы алгоритма объект класса Twist используется для отправки данных с помощью метода publish объекта класса Publisher, который выполняет роль публикации сообщений узлам симуляции (в данном случае сообщения формируются для виртуального транспортного средства).

Блок обратной связи виртуального БСТС предназначен для приема данных сенсорики, установленных на модели виртуального трактора, а также отправки данных на внешнее устройство управления и блок управления. Сообщение обратной связи представляет собой пакет данных, имеющий структуру согласно Таблице 4.2. Структурная схема блока представлена на Рисунке 4.7. Стоит отметить, что видеоданные с камеры напрямую отправляются на внешнее устройство управления.

Таблица 4.2 – Протокол обратной связи виртуального трактора (Протокол пакета #2).

Номера байт	Тип данных	Описание	Данные (диапазон данных)
1	char	префикс	символ '#'
2	char	номер пакета	значение 2
3	int8_t	текущее положение поворота колес	от -100 до 100

Номера байт	Тип данных	Описание	Данные (диапазон данных)
4	int8_t	текущая относительная скорость	от -100 до 100
5	uint8_t	состояние тормоза	0 или 1
6 - 9	uint32_t	показания одометра	весь диапазон типа uint32_t
10	uint8_t	индикатор работы автопилота	0 (автопилот выключен) или 1 (автопилот включен)
11	char	префикс данных сонара	символ '#'
12	uint8_t	номер компоненты пакета	21
13 - 44	float[8]	показания сонаров	массив из восьми вещественных чисел, элементы которого содержат показания сонаров для каждого из каналов
45 - 108	uint64_t	дата и время снятия показаний сонара в миллисекундах	весь диапазон типа uint64_t
109	char	префикс данных сонара	символ '#'
110	uint8_t	номер компоненты пакета	22
111 - 142	float[8]	показания лидара	массив из восьми вещественных чисел, элементы которого содержат показания всех восьми лучей лидара
143 - 206	uint64_t	дата и время снятия показаний лидара в миллисекундах	весь диапазон типа uint64_t
207	char	префикс данных GPS_1	символ '#'
208	uint8_t	номер компоненты пакета	23
209 - 272	uint64_t	дата и время снятия показаний GPS_1 в миллисекундах	весь диапазон типа uint64_t
273 - 276	float	широта GPS_1	от 0.0 до 90.0
277	char	значение полюса GPS_1	символ 'N' -для северной широты, символ 'S' - для южной широты, символ 'M' - для мередиана
278 - 281	float	долгота GPS_1	от 0.0 до 180.0
282	char	значение полюса GPS_1	символ 'E' -для восточной долготы, символ 'W' - для западной долготы, символ 'M' - для мередиана
283 - 286	float	высота GPS_1	диапазон float
287 - 290	float	HDOP GPS_1	диапазон float
291	char	префикс данных GPS_2	символ '#'
292	uint8_t	номер компоненты пакета	24
293 - 296	uint64_t	дата и время снятия показаний GPS_2 в миллисекундах	весь диапазон типа uint64_t
297 - 300	float	широта GPS_2	от 0.0 до 90.0
301	char	значение полюса GPS_2	символ 'N' -для северной широты, символ 'S' - для южной широты, символ 'M' - для мередиана
302 - 305	float	долгота GPS_2	от 0.0 до 180.0
306	char	значение полюса GPS_2	символ 'E' -для восточной долготы, символ 'W' - для западной долготы, символ 'M' - для мередиана
307 - 310	float	высота GPS_2	диапазон float
311 - 314	float	HDOP GPS_2	диапазон float

Номера байт	Тип данных	Описание	Данные (диапазон данных)
315	char	суффикс	символ '*'

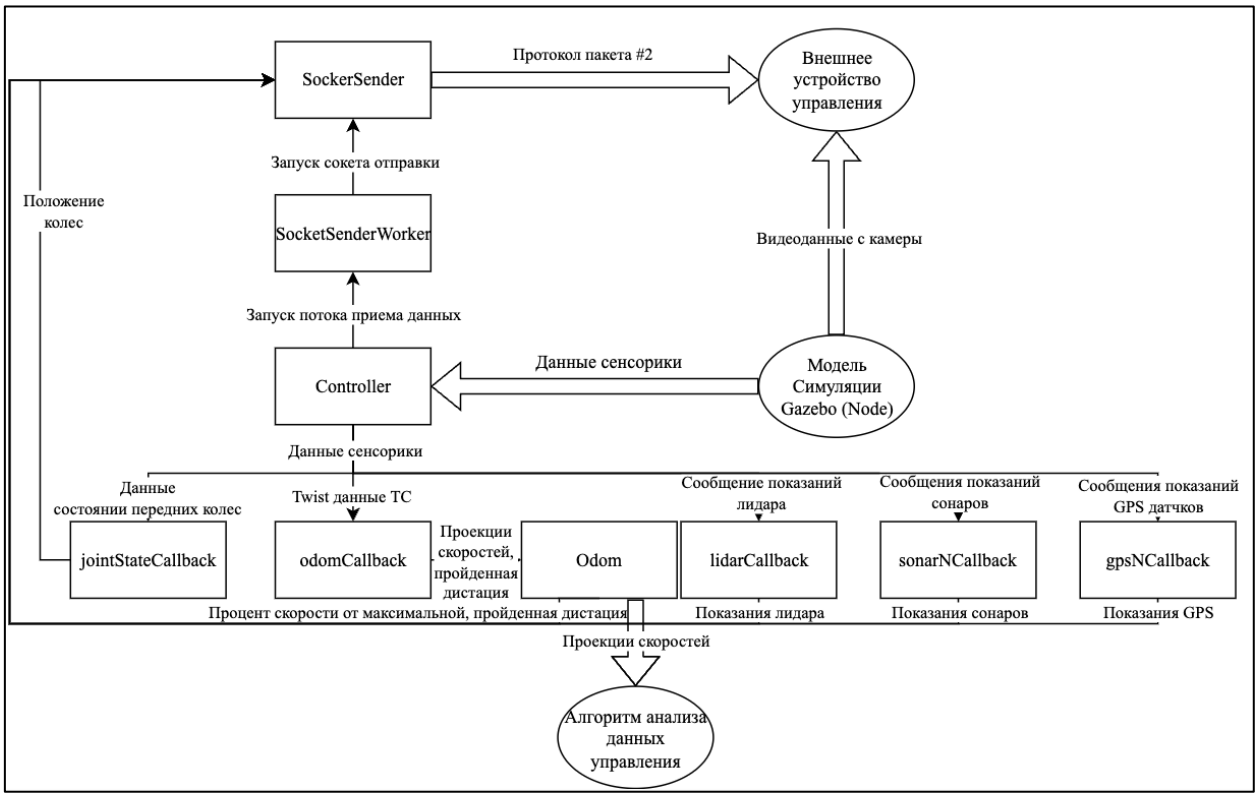


Рисунок 4.7 – Структурная схема блока обратной связи БСТС.

Буфер отправки сокета объекта класса SocketSender каждые 10 мс формирует и отправляет пакет обратной связи, согласно протоколу, указанному в Таблице 4.2, на внешнее устройство управления. Получение данных из других компонент происходит методами реактивного программирования, который предоставляет инфраструктура PyQt. Процедура отправки данных сокетом осуществляется в отдельном потоке, который запускается с помощью объекта класса SocketSenderWorker, инициализация которого происходит в объекте класса Controller.

После запуска симуляции виртуального трактора тяжелого класса описанные в модели датчики и другие компоненты публикуют свои сообщения для подписавшихся клиентов, реализуя модель “Topics

subscription”. В блоке обратной связи объект класса Controller выполняет функцию подписки на предоставляемые симуляцией данные (топики). В разработанной виртуальной симуляцией предоставляются следующие данные:

- Положение колес относительно модели (координаты и углы);
- Скоростные и позиционные характеристики: координаты виртуального транспортного средства и проекции скоростей относительно глобальных координат.
- Показания восьмиканального лидара;
- Показания восьми сонаров;
- Показания двух GPS антенн;

Метод jointStateCallback обрабатывает сообщения о состоянии передних колес. Для каждого переднего колеса выполняется извлечение данных углов поворота колеса. Ввиду того, что поворот колес осуществляется с учетом геометрии рулевого управления Аккермана, угол определяется путем расчета среднего значения относительно правого и левого переднего колеса. Для протокола обратной связи значение угла нормируется в диапазоне от -100 до 100 по формуле (4.3):

$$anglePercent = \frac{meanAngle}{maxSteerAngle} * 100 \quad (4.3),$$

где:

meanAngle – среднее значение угла поворота передних колес,

maxSteerAngle – максимальный угол поворота колес, равный 0.6981 рад.

В результате работы функции испускается QT сигнал с результатом определения нормированного угла поворота колес.

Метод odomCallback выполняет алгоритм по обработке данных модели виртуального трактора объекта типа Twist, получаемых из Gazebo. При обработке из полученных проекций скоростей виртуального ТС вычисляется модуль скорости, а также выполняется запись проекций линейных скоростей по осям X и Y в объект класса Odom. Для протокола обратной связи значение скорости нормируется в диапазон от -100 до 100 по формуле 4.4:

$$velRes = \frac{velMod}{maxSpeed} * 100 \quad (4.4),$$

где:

velMod – модуль скорости виртуального трактора,

maxSpeed – максимальная скорость виртуального трактора, равный 4 м/с.

Так же основе полученных координат выполняется расчет пройденной дистанции. В результате работы функции испускается QT сигнал с результатом определения нормированной скорости и пройденной дистанции. Информация о проекции скоростей так же предоставляется алгоритму анализа данных управления.

Метод lidarCallback обрабатывает показания 8 канального лидара. Симуляция предоставляет данные для каждого угла. Виду этого метод выполняет в первую очередь извлечение данных лучей только на рассматриваемых углах, таким образом формируется массив значений размерностью 8. В случае, если для рассматриваемого луча значение равно “inf”, то для данного элемента массива присваивается значение -1. В результате работы метода испускается QT сигнал с информацией о полученных расстояниях до лучей.

Метод sonarNCallback обрабатывает показания N-ого сонара, где N принимает значение от 1 до 8, то есть для каждого сонара используется свой обработчик. В каждом обработчике выполняется извлечение значений измеренных расстояний. В результате работы метода испускается QT сигнал с информацией о полученном расстоянии до сонара. Если данные расстояния равны “inf”, то используется значение -1.

Метод gpsNCallback обрабатывает показания N-ого датчика GPS, где N принимает значение от 1 до 2, то есть для каждого датчика GPS используется свой обработчик. Для каждого из датчиков из полученных данных извлекаются широта, долгота, их полюсы, высота и HDOP. HDOP получается извлечением корня из суммы квадратов значений из показаний главной диагонали ковариационной матрицы, получаемой в рамках данных датчика,

для осей X и Y. В результате работы метода испускается QT сигнал с вышеописанной информацией.

3. Виртуальное моделирование

3.1 Разработка сценариев виртуального моделирования

Внешнее устройство управления в рамках рассматриваемой системы симуляции виртуального трактора тяжелого класса выполняет роль формирования управляющих пакетов виртуального ТС, обрабатывает пакеты обратной связи, данные камеры и осуществляет работу по следующим сценариям:

- Обработка трактором определенной площади (проезд по указанной площади);
- Объезд препятствия;

Сценарий[118] обработки трактором определенной площади выполняет обработку заданной прямоугольной площади по указанным геодезическим координатам. В рамках указанной площади формируются точки маршрутного задания, по которым виртуальное транспортное средство должно пройти маршрут. На основе точек маршрутного задания и текущего местоположения виртуального ТС формируются управляющие пакеты управления виртуальным трактором согласно протоколу, указанному в таблице 4.1.

Сценарий объезда препятствия выполняет процедуру объезда препятствия в случае его возникновения на пути следования виртуального ТС. В качестве препятствия используется модель человека. Определение препятствия осуществляется анализом данных следующих датчиков:

- Показания сонаров;
- Показания лидара;
- Изображения из видеопотока камеры;

Показания лидара и сонаров входят в пакет обратной связи симуляции виртуального ТС в соответствии с протоколом, указанным в Таблице 4.2. Изображения для анализа системой машинного зрения для определения

препятствия по данным видеопотока камеры напрямую получаются из виртуальной симуляции. Для определения препятствия системой машинного зрения используется обученная модель свёрточной нейросетевой архитектуры YOLO [119-121], которая позволяет определять объекты (в том числе и человека) в режиме реального времени. Для определения человека по данным распознавания используется алгоритм определения расстояния до объекта[122]. Высоту любого объекта можно характеризовать углом обзора камеры этого объекта: чем ближе объект, тем больший угол обзора он занимает, и тем больше его размер на снимке, чем дальше объект, тем меньший угол обзора он занимает и тем меньше его размер на кадре. Расстояние определяется по формуле 4.5.

$$dist = h_p / \tan\psi_{max} * \left(\frac{y_{maxp}}{y_p}\right) \quad (4.5),$$

где:

h_p – средний рост человека,

$\tan\psi_{max} = disH/2F$,

y_{maxp}– половина размера изображения по вертикали,

y_p – рост пешехода в пикселях,

ψ_{max}– половина максимального вертикального угла обзора камеры,

disH – вертикальный размер матрицы камеры,

F – фокусное расстояние.

На Рисунке 4.8 схематично изображена дорожная ситуация с видом сверху, а пешеход находится в вершине какого-либо прямоугольного треугольника, что справа от линзы. Тогда угол φ_1или угол φ_2 искомый. Из треугольников слева от линзы следует: $p_2/p_1 = \tan\varphi_2 / \tan\varphi_1$. Максимизируя φ_1, был получен способ определения угла, показанный в формуле 4.6:

$$\varphi = \tan^{-1}\left(\frac{x_p}{x_{maxp}} * \tan\varphi_{max}\right) \quad (4.6),$$

где:

x_p– отклонение человека от серединной вертикали,

x_{maxp}– половина горизонтального размера изображения,

φ_{max} – половина горизонтального угла обзора.

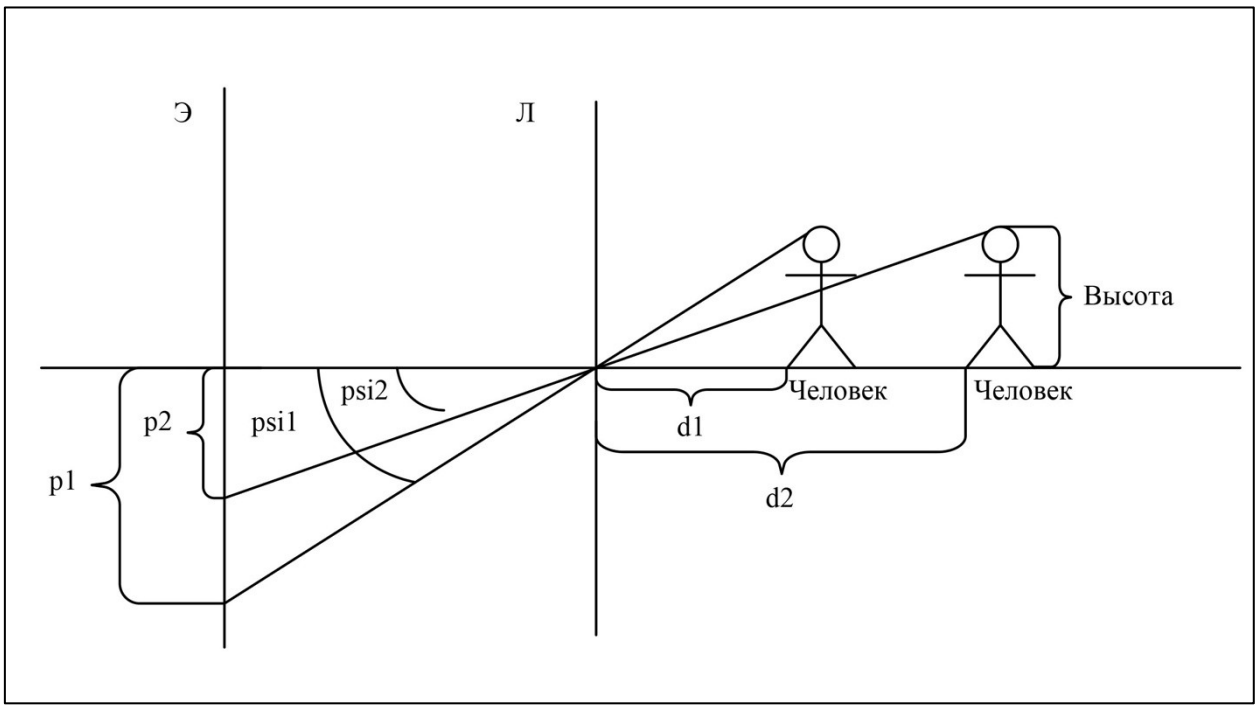


Рисунок 4.8 – Расчет угла до объекта.

На основе полученных данных о препятствии по данным вышеописанной сенсорики определяются точки маршрута для объезда препятствия. На основе точек маршрутного задания и текущего местоположения виртуального ТС формируются управляющие пакеты управления виртуальным трактором согласно протоколу, указанному в Таблице 4.1.

3.2 Описание критериев оценивания результатов виртуального моделирования

Для определения качества выполнения сценариев используются следующие критерии:

- Максимальное отклонение от точек маршрутного задания;
- СКО от точки маршрутного задания;

Формула вычисления максимальных отклонений:

$$max_{dev} = \max(|x_{vi} - x_{ri}|, |y_{vi} - y_{ri}|), i = \overline{1..N} \quad (4.7)$$

где N – количество геолокационных точек;

x_{vi}, y_{vi} - -координаты i-ой точки местоположения БСТС на виртуальном полигоне;

x_{ri}, y_{ri} - -координаты i-ой точки местоположения реального БСТС;

Формула вычисления СКО:

$$\text{СКО} = \sqrt{\frac{1}{N} * \sum_{i=1}^{N}((x_{vi} - x_{ri})^2 + (y_{vi} - y_{ri})^2)} \quad (4.8)$$

где N – количество геолокационных точек;

x_{vi}, y_{vi} - -координаты i-ой точки местоположения ТС на виртуальном полигоне;

x_{ri}, y_{ri} - -координаты i-ой точки местоположения реального ТС.

3.3 Сравнение результатов виртуального моделирования с результатами натурных испытаний

Цель проведения испытаний заключается в верификации алгоритма управления упрощенного автопилота БСТС в процессе движения по маршруту на открытой территории с препятствиями.

Описание:

Подсистемы управления БСТС, система машинного зрения и система виртуального моделирования обменивались информацией по сетевому протоколу. Система управления получала данные о положении колес, скорости движения и направлении движения. Виртуальное БСТС реагировало на команды управления.

Испытания проводились на специальной площадке, где проходила выставка «Всероссийский день поля - 2023»: Республика Татарстан, Лаишевский район, село Большие Кабаны, ул. Выставочная, 1[123].

Полученные результаты:

Были совершены проезды по 5 сценариям. Результаты испытаний в виде графиков представлены на рисунках 4.9-4.13, где зеленым показан маршрут движения реального БСТС, синим цветом – маршрут движения виртуального БСТС на виртуальном полигоне[124]. В Таблице 4.3 представлены результаты испытаний в виде максимальных отклонений (max_{dev}) и средних квадратических отклонений (СКО) маршрутов движения реального и виртуального БСТС друг от друга.

Местоположение реального БСТС фиксировалось с использованием системы GPS/ГНОНАСС с дифференциальными поправками по базовой станции с точностью до 0,3 метра. Местоположение БСТС на виртуальном полигоне определялось однозначно его пространственными координатами в системе виртуального моделирования без погрешности.

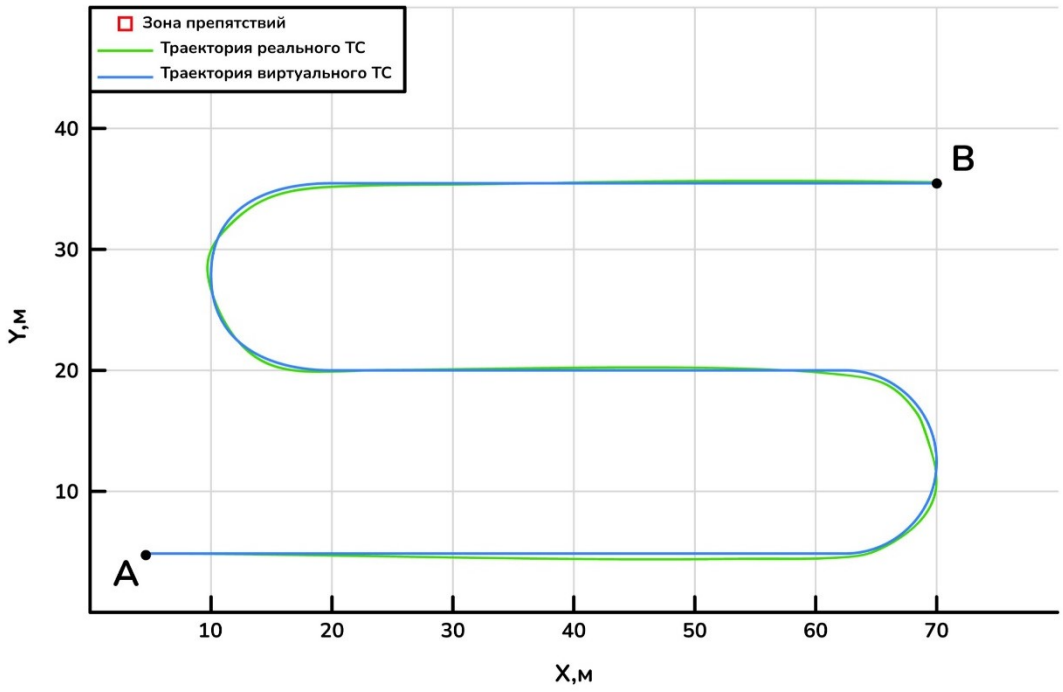


Рисунок 4.9 – Испытание №1

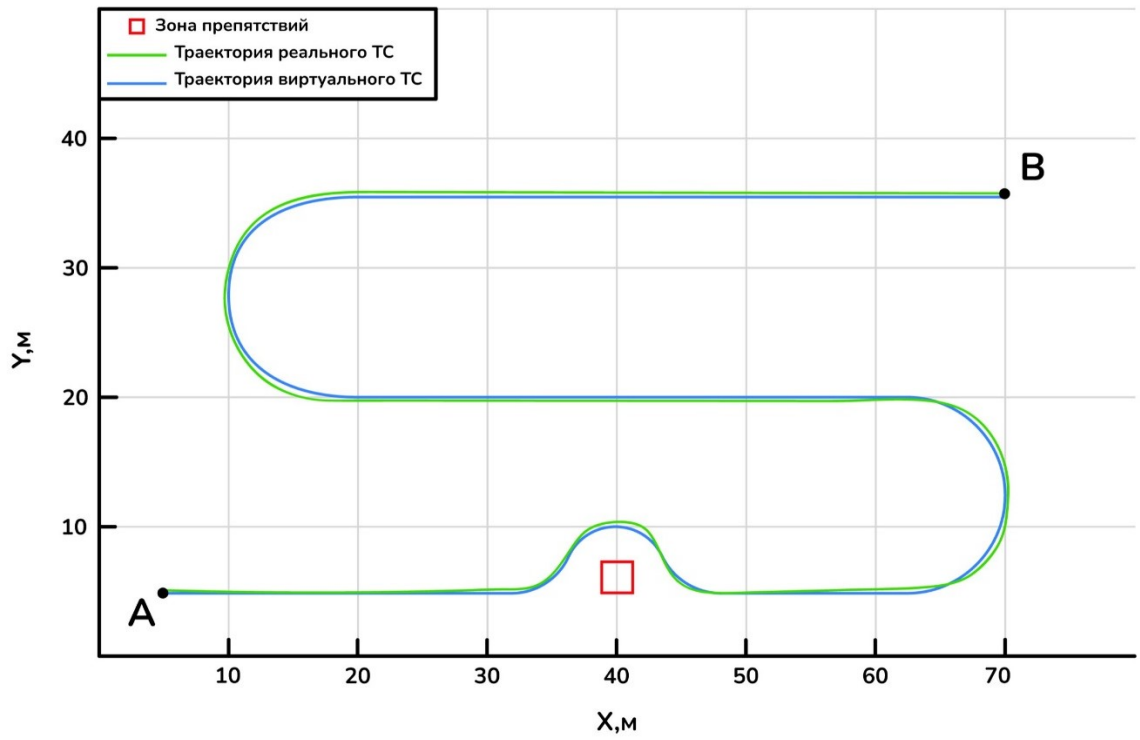


Рисунок 4.10 – Испытание №2

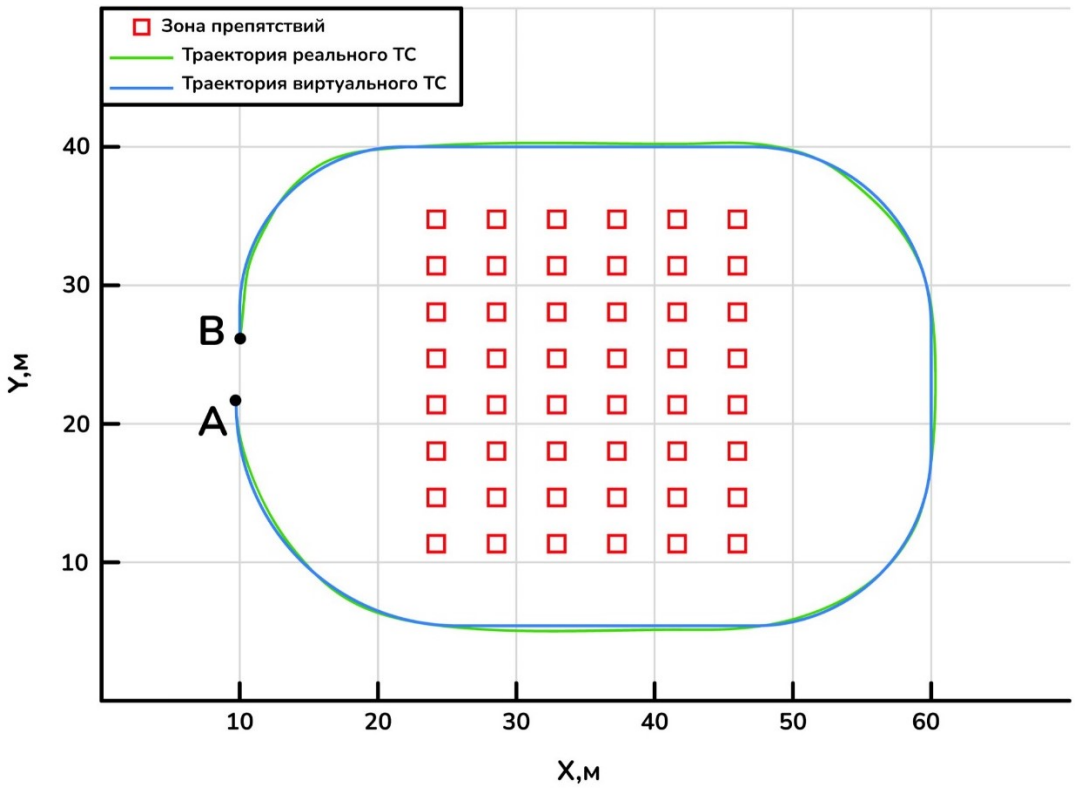


Рисунок 4.11 – Испытание №3

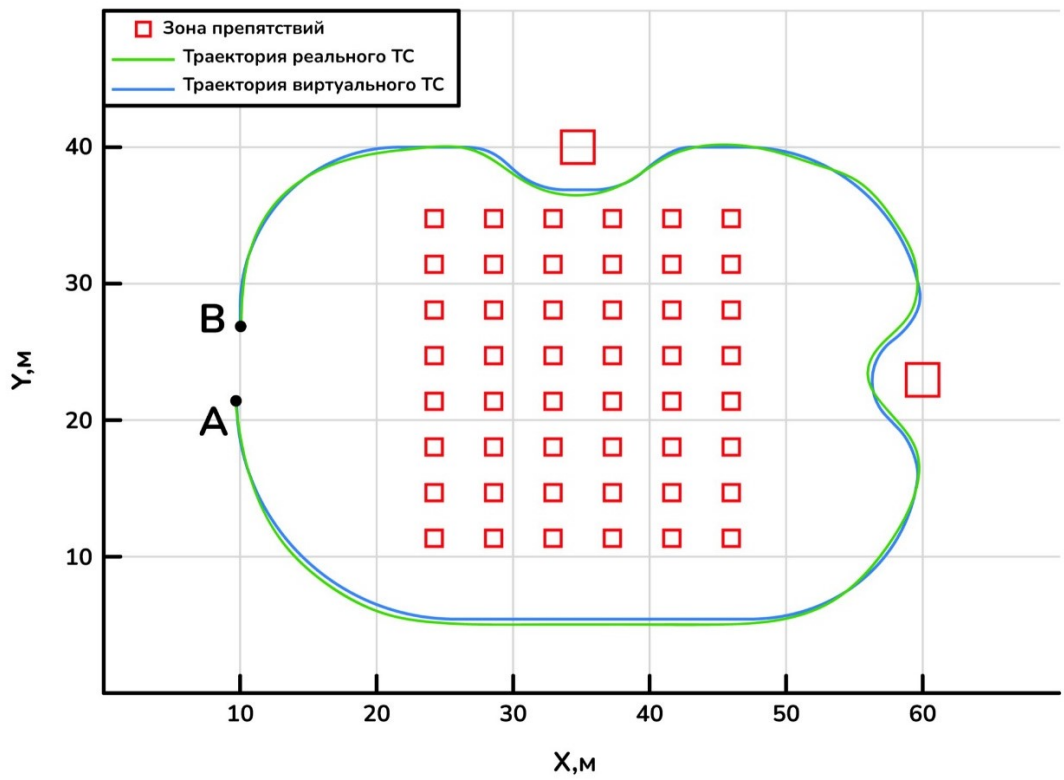


Рисунок 4.12 – Испытание №4

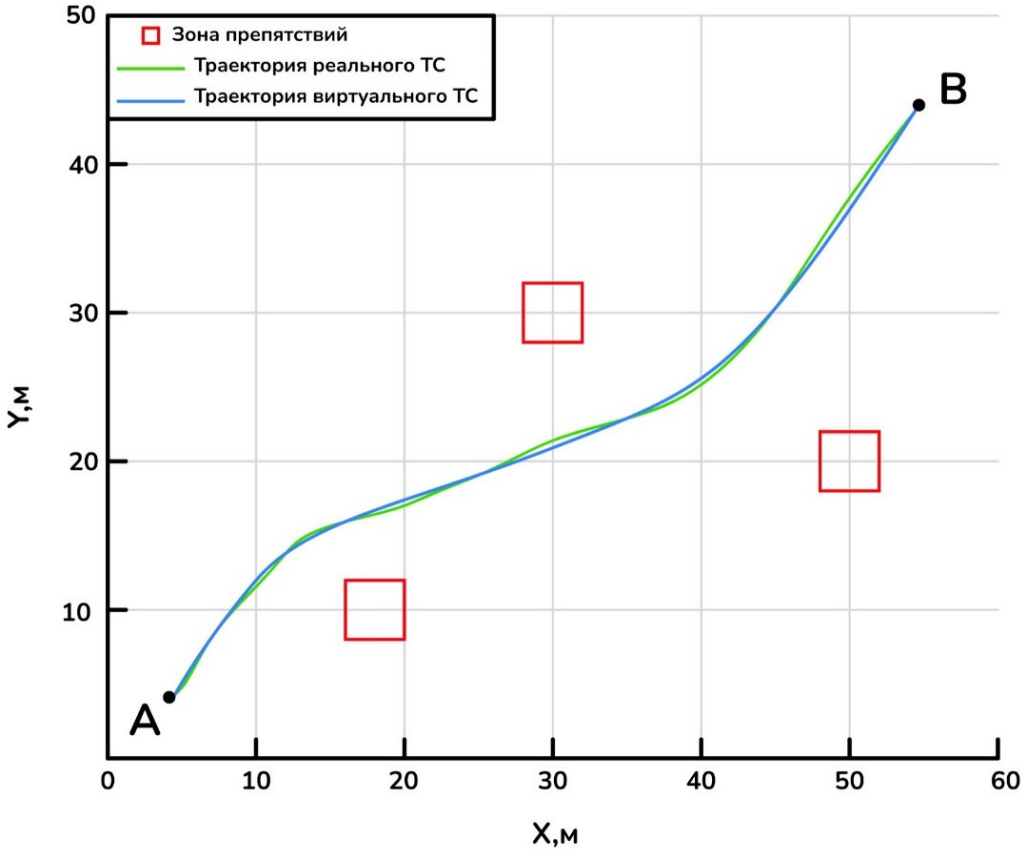


Рисунок 4.13 – Испытание №5

Таблица 4.3 – Результаты испытаний движения реального и виртуального БСТС.

№	Длина маршрута (м)	Максимальное отклонение (м)	СКО (м)
1	154,3	0.324	0.106
2	167,6	0.345	0.109
3	501.6	1.047	0.308
4	544.8	0.905	0.294
5	1506,3	6.321	2.185

По результатам всех проведенных экспериментов наивысшее процентное соотношение максимального отклонения по отношению к общей длине траектории составляет 0,42%, а СКО по отношении к длине траектории: 0,14%. Полученные предельные значения отклонений позволяют говорить о верифицируемости проведенных натурных экспериментов виртуальными испытаниями и высокой достоверности разработанных шаблонов проектирования БСТС.

Заключение главы 4

Глава представляет собой важный этап исследования, сфокусированный на разработке и применении виртуальных моделей для беспилотных сельскохозяйственных транспортных систем (БСТС). Каждый раздел этой главы детально рассматривает аспекты создания виртуальных окружений, моделей транспортных средств, систем сенсорики и алгоритмов управления.

В первом разделе "Постановка задачи виртуального моделирования" была выделена актуальность и сложность задач, связанных с виртуальным моделированием для беспилотных тракторов тяжелого класса. Отмечена необходимость учета оборудования, алгоритмов и систем безопасности для обеспечения эффективности и безопасности в реальных условиях. С использованием программного обеспечения Gazebo и экосистемы ROS2 были выбраны инструменты для создания виртуальных моделей и тестирования автономных систем. Определена важность использования современных технологий виртуального моделирования для обеспечения точности результатов и повышения эффективности разработки.

Во втором разделе "Создание виртуальных моделей" подробно рассматриваются шаги формирования виртуальной окружающей обстановки, виртуального транспортного средства и системы сенсорики с виртуальными датчиками. Особое внимание уделяется разработке алгоритмов управления виртуальным БСТС, обеспечивая реалистичное моделирование процессов управления.

Третий раздел "Виртуальное моделирование" представляет собой применение созданных виртуальных моделей в конкретных сценариях. Здесь проводится разработка сценариев виртуального моделирования, определение критериев оценивания результатов и анализ полученных данных. Результаты виртуального моделирования подвергаются сравнению с результатами натурных испытаний, что позволяет оценить степень согласованности виртуальных и реальных данных.

Важным аспектом является использование виртуального моделирования в качестве инструмента для оптимизации процессов разработки БСТС. Этот метод обеспечивает более быстрое и экономически эффективное тестирование и анализ, предоставляя разработчикам ценные данные для улучшения системы до ее физической реализации.

Таким образом, глава является неотъемлемой частью исследования, обеспечивая тщательное создание и эффективное использование виртуальных моделей в различных аспектах беспилотных сельскохозяйственных транспортных систем.

ЗАКЛЮЧЕНИЕ

Цель данной диссертационной работы заключалась в разработке и внедрении эффективных подходов разработки систем автоматизации для беспилотных сельскохозяйственных транспортных средств (БСТС) с использованием современных методологий и технологий.

В рамках диссертации были получены следующие ключевые результаты:

1. Разработан универсальный шаблон проектирования беспилотных сельскохозяйственных транспортных систем, состоящий из **4** подсистем и **14** функциональных блоков.

2. Определены ключевые системные ограничения информационно-управляющих блоков беспилотной сельскохозяйственной техники:

- ограниченность ресурсов и строгая приоритезация задач **вычислительной подсистемы** (латентность срабатывания на уровне 0,5-1 сек; количество одновременно выполняемых задач не более 2-х).

- пропускная способность, латентность, джиттер, помехоустойчивость **подсистемы коммуникации и связи** (скорость связи без потери соединения не более 1-2 Мбит\с, латентность на уровне 300-400 мс, джиттер на уровне 20-50 мс).

- частота получения навигационных оценок, точность позиционирования, качество сигнала и количество наблюдаемых спутников **в подсистеме позиционирования и навигации** (частота получения навигационных оценок 1Гц, точность позиционирования – 2-3 метра, обеспечивается достаточное качество приема сигналов спутников исключительно на открытой местности, без заезда в укрытия гаражного\ангарного типа).

- скорость обработки данных, количество и качество различных распознаваемых объектов в **подсистеме машинного зрения** (количество обрабатываемых кадров – до 2 в секунду, количества одновременно

анализируемых объектов – от 3 до 5; стабильная работа исключительно в условиях отсутствия засветки и неблагоприятных погодных явлений).

3. Определены методы снятия приведенных системных ограничений:

- вычислительная подсистема: использование операционной системы реального времени совместно с индустриальными микроконтроллерами с приоритезацией задач с использованием расширения реального времени обеспечивает латентность срабатывания на уровне 0,05-0,1 сек с количеством одновременно выполняемых задач – до 14 с динамической регулировкой приоритетов (соответственно количеству функциональных блоков информационно-управляющей системы).

- система позиционирования и навигации: использование многоточечной системы инерциальной навигации одновременно с PPP-позиционированием и взаимной калибровкой показаний обеспечивает повышение частоты обновления навигационных оценок до 100 Гц с уменьшением СКО позиционирования до 0,1 метра. Одновременно с этим достигается функционирование системы навигации в полном автономе (в усло-виях недоступности навигационного созвездия спутников ГНСС) до 10-15 минут с точностью до 1,5% от пройденной траектории в указанном режиме.

- система связи: использование низколатентных профилей радиоинтерфейсов 5G одновременно с применением протокола помехоустойчивой связи без установления соединия позволяет радикально (в 10 и более раз) снизить количество разрывов (по-терь пакетов, сопровождающих повреждения и-или комплексную потерю видеокадра) при скоростях передачи до 10-20 Мбит\с.

- машинное зрение: использование современных аппаратно-акселерированных нейросетей последнего поколения, оптимизированных по размеру видеопамяти и раз-решению совместно с унификацией условий наблюдения при помощи алгоритмов адаптивной нормализации видеопотока обеспечили увеличение показателя количества кадров обработки до 10 в секунду и количества одновременно анализируемых обьектов от 15 до 20, в

том числе при неблагоприятных условиях наблюдения (снег, дождь, туман, сумерки и пр.).

4. Разработана и детально описана методика виртуального проектирования БСТС, приведены аспекты создания виртуальных окружений, моделей транспортных средств, систем сенсорики и алгоритмов управления. По результатам проведенных полунатурных испытаний маршрутов движения реального и виртуального БСТС друг от друга наивысшее процентное соотношение максимального отклонения по отношению к общей длине траектории составляет 0,42%, а СКО по отношении к длине траектории: 0,14%.

Разработанные методологии и принципы автоматизации беспилотных сельскохозяйственных транспортных средств предоставляют значительные выгоды, включая существенное сокращение временных затрат и оптимизацию использования ресурсов в процессе проектирования.

Данные подходы демонстрируют превосходство разработанной системы по ряду ключевых аспектов по сравнению с существующими решениями. Кроме того, система способна функционировать в условиях низкой освещенности и ночного времени, а также в разнообразных метеорологических условиях. Эти результаты успешно приме-нены в проектах по автоматизации тракторов, таких как Беларус-112Н, Беларус-3522.

Обобщая вышеизложенное, результаты исследования не только способствуют развитию теории информационно-управляющих систем сельскохозяйственных транспортных средств, но и имеют прямое практическое применение, внося существенный вклад в современные технологии проектирования и функционирования подобных систем. Полученные научные положения обеспечивают теоретическую основу для дальнейших инноваций в области сельского хозяйства и автономной техники, открывая перспективы для развития сельскохозяйственной отрасли в условиях быстро меняющегося технологического ландшафта.

СПИСОК ЛИТЕРАТУРЫ

1. Интеллектуальные системы в сельском хозяйстве / О. Г. Каратаева, О. В. Виноградов, Д. И. Харламов [и др.] // Научно-информационное обеспечение инновационного развития АПК : материалы XI Международной научно-практической интернет конференции, п. Правдинский, 05–07 июня 2019 года. – п. Правдинский: Российский научно-исследовательский институт информации и технико-экономических исследований по инженерно-техническому обеспечению агропромышленного комплекса, 2019. – С. 268-271. – EDN CQSAQJ.

2. Пономарев, Д. А. Перспективы развития беспилотной сельскохозяйственной техники / Д. А. Пономарев, К. С. Шиянова // Поколение будущего: Взгляд молодых ученых - 2022 : сборник научных статей 11-й Международной молодежной научной конференции, Курск, 10–11 ноября 2022 года. Том 4. – Курск: Юго-Западный государственный университет, 2022. – С. 447-449. – EDN XQOTLQ.

3. Кацуба Ю.Н., Караваев Н.А. Повышение уровня интеграции беспилотных технологий в эксплуатацию сельскохозяйственной техники. Материалы XXXVI Национальной (с Международным участием) научно-технической конференции «Улучшение эксплуатационных показателей и технический сервис автомобилей, тракторов и двигателей», посвященной 95-летию со дня рождения ученых СПбГАУ Буркова Вадима Васильевича, Николаенко Анатолия Владимировича и Кряжкова Валентина Митрофановича. Выпуск № 67 (2023) С. 60-65.

4. Бондарева, А. А. Перспективы применения беспилотного транспорта в России / А. А. Бондарева, Л. Н. Паршина // Научно-техническое и экономическое сотрудничество стран АТР в XXI веке. – 2021. – Т. 1. – С. 127-130. – EDN FJEDRP.

5. Системы точного земледелия на Агропромышленном форуме Юга России-2018. — Текст : электронный // Агропромышленный портал Юга

России : [сайт]. — URL: https://www.agroyug.ru/news/id-29562 (дата обращения: 07.12.2023).

6. CNH INDUSTRIAL. – Текст : электронный // The NewsMarket : [сайт]. – URL: https://preview.thenewsmarket.com/Previews/CNHA/DocumentAssets/511639.pdf (дата обращения: 07.12.2023).

7. Антонов, М. А. Об автоматизации сельского хозяйства / М. А. Антонов, А. А. Анисимов, С. Е. Каширо // Известия Тульского государственного университета. Технические науки. – 2022. – № 9. – С. 210-215. – DOI 10.24412/2071-6168-2022-9-210-215. – EDN AFOVPE.

8. Дисплей GFX-750™. — Текст : электронный // Trimble Agriculture: [сайт]. — URL: https://ru.ptxtrimble.com/product/gfx-750-display/ (дата обращения: 07.12.2023).

9. ТЕХНОЛОГИЯ XFILL™. — Текст : электронный // New Holland Agriculture : [сайт]. — URL: https://agriculture.newholland.com/apac/ru-kz/sistema-upravlenija-tocnym-zemledeliem-plm/produkty/korrektirujusie-istocniki/tehnologija-xfill (дата обращения: 07.12.2023).

10. Система параллельного вождения EZ-Pilot®. — Текст : электронный // Trimble Agriculture : [сайт]. — URL: https://ru.ptxtrimble.com/product/sistema-parallelnogo-vozhdenia-ez-pilot/ (дата обращения: 07.12.2023).

11. Система автовождения Autopilot™ с электроприводом. — Текст : электронный // Trimble Agriculture : [сайт]. — URL: https://ru.ptxtrimble.com/product/sistema-s-electroprivodom-autopilot/ (дата обращения: 07.12.2023).

12. Федоренко, В. Ф. Тенденции цифровизации и интеллектуализации сельского хозяйства / В. Ф. Федоренко // Инновации в сельском хозяйстве. – 2019. – № 1(30). – С. 231-241. – EDN IKJJHN.

13. Горюнова, С. А. Ведение точного сельского хозяйства как способ решения вопроса продовольственной и экономической безопасности нашей

страны / С. А. Горюнова, И. В. Юшин // Базовые тренды социально-экономического развития: вопросы оценки: Материалы региональных научно-практических конференций, Калуга, 26 сентября 2019 года – 08 2020 года. – Калуга: Индивидуальный предприниматель Стрельцов Илья Анатольевич, 2021. – С. 87-100. – EDN NSIUOC.

14. Шуганов, В. М. Основные направления развития цифровизации сельского хозяйства / В. М. Шуганов // Известия Кабардино-Балкарского научного центра РАН. – 2021. – № 2(100). – С. 77-85. – DOI 10.35330/1991-6639-2021-2-100-77-85. – EDN HVUNTZ.

15. Загазежева, О. З. Анализ применения современных роботизированных технологий в сельском хозяйстве и их экономическая эффективность / О. З. Загазежева, С. Х. Шалова, М. А. Канокова // Перспективные системы и задачи управления : Материалы XVII Всероссийской научно-практической конференции и XIII молодёжной школы-семинара, п. Домбай, 04–08 апреля 2022 года. – Таганрог: ИП Марук М.Р, 2022. – С. 289-302. – EDN ZNODTB.

16. Болтовский, С. Н. Беспилотные технологии в сельском хозяйстве / С. Н. Болтовский, Д. В. Панасюк, А. С. Кравец // Инновационные технологии в АПК, как фактор развития науки в современных условиях : Сборник международной научно-исследовательской конференции, посвященной 70-летию создания факультета ТС в АПК (МЕХ ФАК), Омск, 26 ноября 2020 года. – Омск: Омский государственный аграрный университет имени П.А. Столыпина, 2020. – С. 66-70. – EDN ASOWQW.

17. Галиуллин, И. Г. Система автономного управления движением машинно-тракторного агрегата с использованием отечественной элементной базы / И. Г. Галиуллин // Известия Кабардино-Балкарского научного центра РАН. – 2022. – № 6(110). – С. 92-98. – DOI 10.35330/1991-6639-2022-6-110-92-98. – EDN NZIISF.

18. Абросимов В.К. Малые интеллектуальные роботы для решения задач точного земледелия: проблемы и решение / В.К. Абросимов, В.В.

Елисеев // Робототехника и техническая кибернетика. – №4(21). – Санкт-Петербург : ЦНИИ РТК. – 2018. – С. 14-19.

19. Jo, K. Development of Autonomous Car—Part I: Distributed System Architecture and Development Process / J. Kim, D. Kim, C. Jang and M. Sunwoo // IEEE Transactions on Industrial Electronics, vol. 61, no. 12, pp. 7131-7140, Dec. 2014, doi: 10.1109/TIE.2014.2321342.

20. Top Down vs Bottom Up Engineering. — Текст : электронный // Fast Engineering: Working Smarter : [сайт]. — URL: http://www.fastengineering.com.au/single_post.php?id=Top%20Down%20vs%20Bottom%20Up%20Engineering (дата обращения: 07.12.2023).

21. Crespi, V. Top-down vs bottom-up methodologies in multi-agent system design / V. Crespi, A. Galstyan, K. Lerman // Autonomous Robots. – 2008. – Vol. 24, No. 3. – P. 303-313. – DOI 10.1007/s10514-007-9080-5. – EDN TGPXTT.

22. Taming Dr. Frankenstein: Contract-based design for cyber-physical systems / Sangiovanni-Vincentelli A., Damm W., Passerone R. // European journal of control. — 2012. — Vol. 18, No. 3. — P. 217-238. — DOI: 10.3166/ejc.18.217-238.

23. Heinecke H. Automotive system design – challenges and potential // Proceedings. Design, Automation and Test in Europe. Munich, Germany. March 7 – 11, 2005. — Vol. 1. — Los Alamitos: IEEE Computer Society, 2005. — P. 656-657. — DOI: 10.1109/DATE.2005.79.

24. Matthaei, R. and Maurer, M. (2015) Autonomous driving – a top-down-approach. at — Automatisierungstechnik, Vol. 63 (Issue 3), pp. 155-167 — DOI: 10.1515/auto-2014-1136.

25. Xiongfeng Feng. Enhanced supervisory control system design of an unmanned ground vehicle / P. C. Y. Chen, A. N. Poo, J. I. Guzman and C. W. Chan // 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583), The Hague, Netherlands, 2004, pp. 1864-1869, doi: 10.1109/ICSMC.2004.1399935.

26. Forsberg K., Mooz H. System Engineering for Faster, Cheaper, Better // INCOSE International Symposium. — 1999. — Vol. 9, no. 1. — P. 924-932. — DOI: 10.1002/j.2334-5837.1999.tb00258.x

27. Чикрин, Д. Е. Методология S.M.A.R.T.E.S.T. H-GQM для контролируемой эволюции систем ADAS / Д. Е. Чикрин, А. А. Егорчев, Д. В. Ермаков // Известия ЮФУ. Технические науки. – 2020. – № 2(212). – С. 200-209. – DOI 10.18522/2311-3103-2020-2-200-209. – EDN MWLOMV.

28. Чикрин, Д. Е. МЕТОДОЛОГИЧЕСКИЕ ОСНОВЫ ПРОЕКТИРОВАНИЯ ИНФО-КОММУНИКАЦИОННЫХ СИСТЕМ АВТОМОБИЛЬНЫХ ТРАНСПОРТНЫХ СРЕДСТВ ВЫСОКОЙ СТЕПЕНИ АВТОМАТИЗАЦИИ : дис. … д-ра техн. Наук, 05.13.01. – Казань, 2021. – 399с.

29. Егорчев А.А. ВЕРИФИЦИРУЕМЫЕ СИСТЕМЫ ВИРТУАЛЬНОГО МОДЕЛИРОВА-НИЯ БЕСПИЛОТНЫХ ТРАНСПОРТНЫХ СРЕДСТВ : дис. … канд. техн. наук, 05.13.01. – Казань, 2021. – 340с.

30. James, P. W. The Machine That Changed the World / James P. Womack, Daniel T. Jones, Daniel Roos — New York : FREE PRESS, 2007. — 352 с. — Текст : непосредственный.

31. What is Lean? — Текст : электронный // Lean Enterprise Institute : [сайт]. — URL: https://www.lean.org/explore-lean/what-is-lean/ (дата обращения: 07.12.2023).

32. Bonamigo, A. Lean thinking in value co-creation: a basis for innovation in agroindustrial sector / Corrêa, A. G. de A.; de Oliveira, G. S. C.; da Silva, P. V. de S. C.; Andrade, H. de S. // International Journal of Innovation. — 2023. — Vol. 11, no. 3. — p. e24034. — DOI: 10.5585/2023.24034.

33. Kaizen. — Текст : электронный // Lean Enterprise Institute : [сайт]. — URL: https://www.lean.org/lexicon-terms/kaizen/ (дата обращения: 07.12.2023).

34. 5S. — Текст : электронный // Lean Enterprise Institute : [сайт]. — URL: https://www.lean.org/lexicon-terms/five-s/ (дата обращения: 07.12.2023).

35. Value Stream Mapping. — Текст : электронный // Lean Enterprise Institute : [сайт]. — URL: https://www.lean.org/lexicon-terms/value-stream-mapping/ (дата обращения: 07.12.2023).

36. Голдратт, Элияху Цель. Процесс непрерывного улучшения. / Элияху Голдратт, Джефф Кокс. — Минск : Поппури, 2021. — 400 с. — Текст : непосредственный.

37. Michal, M. and Iveta, P. Applying the Theory of Constraints in the Course of Process Improvement. // Research Papers Faculty of Materials Science and Technology Slovak University of Technology. — 2010. — Vol. 18, no. 29. — p. 71-76. — DOI: 10.2478/v10186-010-0028-9.

38. Real-time operating system for microcontrollers and small microprocessors. — Текст : электронный // FreeRTOS™ : [сайт]. — URL: https://www.freertos.org/ (дата обращения: 07.02.2024).

39. Артамонов, Н. С. Классификация дорожных знаков при помощи YOLO CNN на мобильной платформе NVIDIA Jetson / Н. С. Артамонов, П. Ю. Якимов // Информационные технологии и нанотехнологии : Сборник трудов ИТНТ-2018, Самара, 24–27 апреля 2018 года / Самарский национальный исследовательский университет имени академика С.П. Королева. – Самара: Предприятие "Новая техника", 2018. – С. 2328-2334. – EDN XMXAOD.

40. Золотарев, С. Операционные системы реального времени для 32-разрядных микропроцессоров / С. Золотарев. — Текст : непосредственный // СОВРЕМЕННАЯ ЭЛЕКТРОНИКА. — 206. — № 7. — С. 52-59.

41. МТС выпустила комплекты 5G-оборудования для создателей цифровых решений. — Текст : электронный // МТС : [сайт]. — URL: https://moskva.mts.ru/about/media-centr/soobshheniya-kompanii/novosti-mts-v-rossii-i-mire/2022-10-04/mts-vypustila-komplekty-5g-oborudovaniya-dlya-sozdatelej-cifrovyh-reshenij (дата обращения: 07.02.2024).

42. Разработка высокоточной спутниковой локально-инерциальной системы навигации для беспилотного управления транспортными средствами / Д. Е. Чикрин, П. А. Савинков, П. А. Кокунин, Р. И. Шагиев // Известия

высших учебных заведений. Приборостроение. – 2020. – Т. 63, № 12. – С. 1094-1102. – DOI 10.17586/0021-3454-2020-63-12-1094-1102. – EDN VJVIKN.

43. Immerkaer J. Fast Noise Variance Estimation // Computer Vision and Image Understanding. 1996. Vol. 2. P. 300-302.

44. Sharpness Estimation From Image Gradients. — Текст : электронный // MathWorks : [сайт]. — URL: https://www.mathworks.com/matlabcentral/fileexchange/32397-sharpness-estimation-from-image-gradients (дата обращения: 04.02.2024).

45. Автоматическая оценка и предобработка изображений видеопотока для задач машинного зрения / Чикрин Д.Е., Малюгина А.А., Державин Д.В., Егорчев А.А. // Современная наука: актуальные проблемы теории и практики. Серия: естественные и технические науки. — 2018. — №6. — С. 145-157.

46. Державин Д. В. Система машинного зрения на базе архитектуры параллельных вычислений на графическом процессоре Jetson TX2 : магистерская дисс. — Казанский (Приволжский) федеральный ун-т, Казань, 2018. — С. 24-35. — URL: https://kpfu.ru/student_diplom/10.160.178.20_6174008_VKR_Derzhavin.pdf (дата обращения: 04.02.2024).

47. Задачи управления движением автономных колесных роботов в точном земледелии / Т. А. Тормагов, А. А. Генералов, М. Ю. Шавин, Л. Б. Рапопорт // Гироскопия и навигация. – 2022. – Т. 30, № 1(116). – С. 39-60. – DOI 10.17285/0869-7035.0083. – EDN GWCIQX.

48. Свидетельство о государственной регистрации программы для ЭВМ № 2021667661 Российская Федерация. Программный модуль сегментации объектов препятствий дерево и столб на основе нейронных сетей : № 2021666528 : заявл. 20.10.2021 : опубл. 01.11.2021 / Р. Ф. Сабиров, А. Р. Валиев, В. М. Медведев [и др.] ; заявитель ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ». – EDN MBAXKT.

49. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М.: Техносфера, 2005. – 1072 с.

50. Сикорский, О.С. Обзор свёрточных нейронных сетей для задачи классификации изображений / О.С. Сикорский // Новые информационные технологии в автоматизированных системах – Москва, 2017. – №20. – С. 37–42.

51. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение – СПб.: Питер, 2018. – 480 с.

52. R. R. Atole, D. Park, A Multiclass Deep Convolutional Neural Network Classifier for Detection of Common Rice Plant Anomalies / (IJACSA) International Journal of Advanced Computer Science and Applications. – Vol. 9, No. 1. – 2018. – P. 67-70.

53. R.Rajmohan, M.Pajany, R.Rajesh, D.Raghu Raman, U. Prabu, Smart paddy crop disease identification and management using deep convolution neural network and SVM classifier / International Journal of Pure and Applied Mathematics. – Volume 118 No. 15. – 2018. – P. 255-264.

54. G. Athanikar1, P. Badar, Potato Leaf Diseases Detection and Classification System / International Journal of Computer Science and Mobile Computing. – Vol.5 Issue.2. – 2016. – P. 76-88.

55. S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification / Computational Intelligence and Neuroscience. – 2016. – 11 p.

56. Abdullahi, H.S. Advances of image processing in Precision Agriculture: Using deep learning convolution neural network for soil nutrient classification / Halimatu Sadiyah. Abdullahi, Oba Mustpha Zubair // Journal of Multidisciplinary Engineering Science and Technology (JMEST) – Vol. 4 Issue 8, August – 2017. – P 7981-7987.

57. Huang H., Deng J., Lan Y., Yang A., Deng X., Zhang L. A fully convolutional network for weed mapping of unmanned aerial vehicle (UAV) imagery / PLoS ONE 13(4): e0196302. – 2018.

58. Inkyu Sa, weedNet: Dense Semantic Weed Classification Using Multispectral Images and MAV for Smart Farming / Inkyu Sa, Zetao Chen, Marija Popovic, Raghav Khanna, Frank Liebisch, Juan Nieto, Roland Siegwa // IEEE Robotics and Automation Letters. – 2018 – Vol. 3(1). – P. 588-595.

59. Potena C., Nardi D., Pretto A. Fast and Accurate Crop and Weed Identification with Summarized Train Sets for Precision Agriculture / IAS 2016: Intelligent Autonomous Systems 14. – 2017. – P 105-121

60. Yao, C., Zhang, Y., Zhang, Y., and Liu, H.: Application of convolutional neural network in classification of high resolution agricultural remote sensing images, Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XLII2/W7. – 2017. – P. 989-992.

61. R. R. Atole, D. Park, A Multiclass Deep Convolutional Neural Network Classifier for Detection of Common Rice Plant Anomalies / (IJACSA) International Journal of Advanced Computer Science and Applications. – Vol. 9, No. 1. – 2018. – P. 67-70.

62. Ganchenko, V. Image Semantic Segmentation Based on Convolutional Neural Networks for Monitoring Agricultural Vegetation / V. Ganchenko, A. Doudkin // Communications in Computer and Information Science, Springer, 2019. – Ch. 5. – V. 1055. – 2019. – P. 52-63.

63. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. / In Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, 2015.

64. V. Badrinarayanan, A. Kendall and R. Cipolla, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation / IEEE Transactions on Pattern Analysis and Machine Intelligence – vol. 39 no. 12 – 1 Dec. 2017 – pp. 2481-2495 – doi: 10.1109/TPAMI.2016.2644615.

65. G. Huang, Z. Liu, L. Van Der Maaten and K. Weinberger, Densely Connected Convolutional Networks / IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017 pp. 2261-2269. doi: 10.1109/CVPR.2017.243

66. Ганченко, В. В. Распознавание сельскохозяйственной растительности на изображениях земной поверхности на основе сверточной нейронной сети U-Net / В. В. Ган-ченко, А. А. Дудкин, С. В. Шелег // Big Data and Advanced Analytics. – 2021. – № 7-1. – С. 110-116. – EDN ZDCYJM

67. Kots, M. V. U-Net adaptation for multiple instance learning / M. V. Kots, V. S. Chukanov // Journal of Physics: Conference Series, Saint Petersburg, 21–22 марта 2019 года. Vol. 1236. – Saint Petersburg: Institute of Physics Publishing, 2019. – P. 012061. – DOI 10.1088/1742-6596/1236/1/012061. – EDN SJUOKU.

68. Валиев, А. Р. Беспилотный трактор / А. Р. Валиев, Мануель Бинело, Б. Г. Зиганшин, Р. Ф. Сабиров, Г. Т. Шафигуллин, И. Г. Галиуллин // Вестник НЦБЖД. – 2021. – № 4 (50). – С.69–75.

69. What Is CUDA?. — Текст : электронный // NVIDIA : [сайт]. — URL: https://blogs.nvidia.com/blog/what-is-cuda-2/ (дата обращения: 04.02.2024).

70. Мини-трактор Беларус-112Н-01. — Текст : электронный // MTZ-TATRSTAN : [сайт]. — URL: https://www.mtz-tatarstan.ru/catalog/traktora/mini-tehnika-i-malogabaritnye-traktora/100-seriya/mini-traktor-belarus-112n-01-9d95c4c3.html (дата обращения: 14.02.2024).

71. Трактор "Беларус 3522-39/131-46/461" (ОАО МТЗ) дв.Камминз. — Текст : электронный // MTZ-TATRSTAN : [сайт]. — URL: https://www.mtz-tatarstan.ru/catalog/traktora/traktory-obcshego-naznacheniya/3500-seriya/traktor-belarus-3522-39131-46461-oao-mtz-dvkamminz-9d95c4e7.html (дата обращения: 14.02.2024).

72. Свидетельство о государственной регистрации программы для ЭВМ № 2021667314 Российская Федерация. Программный модуль поиска

энергетических максимумов координатных переходов : № 2021666527 : заявл. 20.10.2021 : опубл. 27.10.2021 / Р. Ф. Сабиров, А. Р. Валиев, В. М. Медведев [и др.] ; заявитель ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ». – EDN USHDZC.

73. Свидетельство о государственной регистрации программы для ЭВМ № 2021667402 Российская Федерация. Программный модуль коррекции движения машинно-тракторного агрегата для движения параллельно обработанному участку поля : № 2021666850 : заявл. 20.10.2021 : опубл. 28.10.2021 / Р. Ф. Сабиров, А. Р. Валиев, В. М. Медведев [и др.] ; заявитель ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ». – EDN VXOFLB.

74. Свидетельство о государственной регистрации программы для ЭВМ № 2021667403 Российская Федерация. Программное обеспечение удаленного управления машинно-тракторным агрегатом : № 2021666842 : заявл. 20.10.2021 : опубл. 28.10.2021 / Р. Ф. Сабиров, А. Р. Валиев, В. М. Медведев [и др.] ; заявитель ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ». – EDN ALSSYD.

75. Свидетельство о государственной регистрации программы для ЭВМ № 2021667404 Российская Федерация. Программный модуль распознавания борозды обротанного поля посредством оптического лидара : № 2021666837 : заявл. 20.10.2021 : опубл. 28.10.2021 / Р. Ф. Сабиров, А. Р. Валиев, В. М. Медведев [и др.] ; заявитель ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ». – EDN KBLQBE.

76. Свидетельство о государственной регистрации программы для ЭВМ № 2021667448 Российская Федерация. Программный модуль управления механическими узлами машинно-тракторного агрегата : № 2021666526 : заявл. 20.10.2021 : опубл. 29.10.2021 / Р. Ф. Сабиров, А. Р. Валиев, В. М. Медведев [и др.] ; заявитель ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ». – EDN MJAERL.

77. Свидетельство о государственной регистрации программы для ЭВМ № 2021667510 Российская Федерация. Программный модуль построения маршрута машинно-тракторного агрегата : № 2021666903 : заявл. 25.10.2021 : опубл. 29.10.2021 / Р. Ф. Сабиров, А. Р. Валиев, В. М. Медведев [и др.] ; заявитель ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ». – EDN SNGFTN.

78. Свидетельство о государственной регистрации программы для ЭВМ № 2021667845 Российская Федерация. Программный модуль распознавания борозды обработанного поля посредством камеры видимого диапазона : № 2021667021 : заявл. 26.10.2021 : опубл. 03.11.2021 / Р. Ф. Сабиров, А. Р. Валиев, В. М. Медведев [и др.] ; заявитель ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ». – EDN LYNKWM.

79. Свидетельство о государственной регистрации программы для ЭВМ № 2021668192 Российская Федерация. Программный модуль объезда единичного препятствия машинно-тракторного агрегата : № 2021666619 : заявл. 20.10.2021 : опубл. 10.11.2021 / Р. Ф. Сабиров, А. Р. Валиев, В. М.

Медведев [и др.] ; заявитель ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ». – EDN VYTETE.

80. Свидетельство о государственной регистрации программы для ЭВМ № 2021668265 Российская Федерация. Программный модуль движения машинно-тракторного агрегата по заданным спутниковым координатам по модели схождения азимутов : № 2021666617 : заявл. 20.10.2021 : опубл. 11.11.2021 / Р. Ф. Сабиров, А. Р. Валиев, В. М. Медведев [и др.] ; заявитель ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ». – EDN RXQEBQ.

81. Свидетельство о государственной регистрации программы для ЭВМ № 2021668266 Российская Федерация. Программный модуль деления облака точек поверхности на слои : № 2021666615 : заявл. 20.10.2021 : опубл. 11.11.2021 / Р. Ф. Сабиров, А. Р. Валиев, В. М. Медведев [и др.] ; заявитель ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ». – EDN TJDTBI.

82. Программно-аппаратный комплекс сегментации препятствий с архитектурой U-Net для автономной сельскохозяйственной техники / И. Г. Галиуллин, Р. Ф. Сабиров, Д. Е. Чикрин, А. А. Егорчев // Известия ЮФУ. Технические науки. – 2023. – № 3(233). – С. 46-55. – DOI 10.18522/2311-3103-2023-3-46-55. – EDN TEXLOR.

83. SK - СЕРВИСБЛОК «КОСМОС» SKPM.ON, SKPM.LS, SKPM.LS.Ф. — Текст : электронный // Гидроруль : [сайт]. — URL: https://gidrorul.ru/news/novaya-kollektsiya-puma-sozdana-udivlyat/ (дата обращения: 18.02.2024).

84. Свидетельство о государственной регистрации программы для ЭВМ № 2022665990 Российская Федерация. Программа низкоуровневого управления системы "Беспилотный трактор КФУ-МТЗ-112" : № 2022661910 : заявл. 27.06.2022 : опубл. 24.08.2022 / Д. Е. Чикрин, А. А. Егорчев, И. Г. Галиуллин [и др.] ; заявитель федеральное государственное автономное образовательное учреждение высшего образования «Казанский. – EDN IADXJQ.

85. Свидетельство о государственной регистрации программы для ЭВМ № 2022665171 Российская Федерация. Программа низкоуровневой обработки данных сенсорики лидарного типа системы "Беспилотный трактор КФУ-МТЗ-112" : № 2022661890 : заявл. 27.06.2022 : опубл. 11.08.2022 / Д. Е. Чикрин, А. А. Егорчев, И. Г. Галиуллин [и др.] ; заявитель федеральное государственное автономное образовательное учреждение высшего образования «Казанский. – EDN BPQNUL.

86. YOLO target recognition model - DarkNet53 network structure. — Текст : электронный // ProgrammerSought : [сайт]. — URL: https://programmersought.com/article/56568989475/ (дата обращения: 18.02.2024).

87. Ultralytics YOLOv8. — Текст : электронный // Ultralytics YOLO Docs : [сайт]. — URL: https://docs.ultralytics.com/models/yolov8/ (дата обращения: 18.02.2024).

88. Vasilyeva, Natalia & Korol, Roman. Theoretical issues of the organization of transportation using unmanned vehicles. / Innotrans. - 2023. - 3-6. 10.20291/2311-164X-2023-2-3-6.

89. Особенности моделирования и оценка эффективности производственно-технических систем на ранних стадиях проектирования / А. Р. Ротт, С. Я. Алибеков, А. В. Маряшев [и др.] // Вестник Казанского технологического университета. – 2014. – Т. 17, № 7. – С. 308-314. – EDN SCNMFT.

90. Вопросы достоверного физического и визуального моделирования сложных технических систем беспилотной сельскохозяйственной техники на базе трактора Беларус-3525 / И.Г. Галиуллин, П.А. Тимершин // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки. – 2024. – № 8. – С. 83-87. – DOI 10.37882/2223-2966.2024.8.13.

91. Гайнетдинов, А. Ф. Обзор и сравнение коммерческих и открытых программных комплексов для моделирования робототехнических систем / А. Ф. Гайнетдинов // Молодежный научно-технический вестник. – 2015. – № 9. – С. 7. – EDN ULZMXN.

92. Абдурашитов, А. И. обзор виртуальных сред для тестирования беспилотных транспортных средств / А. И. Абдурашитов, И. С. Аблялимов // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. – 2022. – № 1(35). – С. 5-10. – EDN VHRATQ.

93. Henle, Jacqueline & Stoffel, Martin & Schindewolf, Marc & Nägele, Ann-Therese & Sax, Eric. Architecture platforms for future vehicles: a comparison of ROS2 and Adaptive AUTOSAR. - (2022). - 3095-3102. 10.1109/ITSC55140.2022.9921894.

94. Alexovič, Stanislav & Lacko, Milan & Bačík, Ján. Simulation of Multiple Autonomous Mobile Robots Using a Gazebo Simulator. - (2023). - 10.1007/978-3-031-21435-6_30.

95. Platt, Jonathan & Ricks, Kenneth. Comparative Analysis of ROS-Unity3D and ROS-Gazebo for Mobile Ground Robot Simulation / Journal of Intelligent & Robotic Systems. - (2022). - 106. 10.1007/s10846-022-01766-2.

96. Шабалина, К. С. Виртуальный подход для проведения автоматизированных экспериментов сравнения систем координатных меток в среде Gazebo / К. С. Шабалина, А. Г. Сагитов, Е. А. Магид // Вестник НЦБЖД. – 2018. – № 1(35). – С. 136-143. – EDN YUUDWG.

97. SDFormat Specification. — Текст : электронный // SDFormat : [сайт]. — URL: http://sdformat.org/spec (дата обращения: 18.03.2024).

98. Применение современных технологических решений для создания виртуальных моделей реального мира / А. М. Рыбкина, М. А. Кравцова, К. А. Беляева [и др.] // Международный журнал прикладных наук и технологий Integral. – 2023. – № 4. – EDN UECNUC.

99. Cogo, Emir & Cogo, Ehlimana & Prazina, Irfan & Becirovic, Seila & Okanovic, Vensada & Rizvić, Selma & Mulahasanović, Razija. A Survey of Procedural Modelling Methods for Layout Generation of Virtual Scenes / Computer Graphics Forum. - (2023). - 43. 10.1111/cgf.14989.

100. Zhang, D. & Zhu, X. 3D modeling algorithm based on dynamic terrain grid rendering / Revista de la Facultad de Ingenieria. - (2017). - 32. 469-476.

101. Виртуальное физическое и визуальное моделирование работы механических элементов технических систем / Д. В. Державин, А. А. Егорчев, И. Е. Свалова, Д. Е. Чикрин // Перспективы науки. – 2018. – № 3(102). – С. 25-32. – EDN XPLUYP.

102. Podrigalo, Mikhail & Razarenov, Leonid & Zakapko, Oleksandr. ASSESSMENT OF THE STABILITY OF THE FRONT PIVOT AXLE DURING STEADY-STATE MOTION OF A TRACTOR SELF-PROPELLED CHASSIS / Bulletin of the National Technical University «KhPI» Series Engineering and CAD. - (2023). - 76-81. 10.20998/2079-0775.2023.1.08.

103. Разделение движений на "быстрые" и "медленные" при построении динамических моделей транспортных средств / С. Е. Манянин, П. Е. Дмитриев, Ю. И. Палутин [и др.] // Труды НГТУ им. Р.Е. Алексеева. – 2018. – № 4(123). – С. 227-232. – DOI 10.46960/1816-210X_2018_4_227. – EDN YRRYNV.

104. Кочнев, В. А. Особенности моделирования движения транспортных средств для построения беспилотных систем / В. А. Кочнев, А. А. Локтев // Внедрение современных конструкций и передовых технологий в путевое хозяйство. – 2022. – Т. 18. – С. 96-105. – EDN VCNQIB.

105. Иоффе, М. Л. Принцип Аккермана и его реализация в современных автомобилях / М. Л. Иоффе // ИЗВЕСТИЯ ВЫСШИХ УЧЕБНЫХ

ЗАВЕДЕНИЙ. МАШИНОСТРОЕНИЕ – 2021. – № 9(738) – С. 40-47. – DOI 10.18698/0536-1044-2021-9-40-47.

106. Sui, Donghao & Zhang, Yulai. Analyzation of the Application Scenarios of Ackerman Geometry based on Vehicle Steering Model / Highlights in Science, Engineering and Technology. - (2023). - 46. 71-82. 10.54097/hset.v46i.7676.

107. Shih, Chi-Huang & Lin, Cheng-Jian & Jhang, Jyun-Yu. Ackerman Unmanned Mobile Vehicle Based on Heterogeneous Sensor in Navigation Control Application. / Sensors. - (2023). - 23. 4558. 10.3390/s23094558.

108. Акжигитов, Р. Р. Исследование способов применения компьютерного зрения при моделировании поведения беспилотного автомобиля в виртуальном пространстве / Р. Р. Акжигитов, А. Ю. Политов, К. А. Судариков // Инновации. Наука. Образование. – 2021. – № 28. – С. 1025-1038. – EDN SSBBWE.

109. Gökçe, Barış & Aslan, Y & Koca, Yavuz. Simulation of A Skid Steer Driving Mobile Robot in ROS / Gazebo Environment. - (2020). - 1. 29-41.

110. Nowakowski, Marek & Kurylo, Jakub. Usability of Perception Sensors to Determine the Obstacles of Unmanned Ground Vehicles Operating in Off-Road Environments. / Applied Sciences. - (2023). - 13. 4892. 10.3390/app13084892.

111. Zhao, Zixu & Zhang, Yucheng & Long, Long & Lu, Zaiwang & Shi, Jinglin. Efficient and adaptive lidar–visual–inertial odometry for agricultural unmanned ground vehicle. / International Journal of Advanced Robotic Systems. - (2022). - 19. 172988062210949. 10.1177/17298806221094925.

112. Чикрин, Д. Е. Логика и структура построения системы управления беспилотных транспортных средств / Д. Е. Чикрин // Известия Самарского научного центра Российской академии наук. – 2021. – Т. 23, № 4(102). – С. 96-102. – DOI 10.37313/1990-5378-2021-23-4-96-102. – EDN JORJHX.

113. Румянцев, Ю. А. Идентификация модели управления колесного робота в симуляторе Gazebo нейронной сетью / Ю. А. Румянцев // Вопросы

теории безопасности и устойчивости систем. – 2021. – № 23. – С. 40-48. – EDN YRUIEL.

114. Прокопьев, И. В. Исследование метода идентификации модели и методов управления беспилотным транспортным средством по пространственной траектории / И. В. Прокопьев, Е. А. Софронова // Надежность и качество сложных систем. – 2020. – № 3(31). – С. 99-111. – DOI 10.21685/2307-4205-2020-3-12. – EDN FBJJWJ.

115. Chen, Yimin & Hu, Chuan & Qin, Yechen & Li, Mingjun & Song, Xiaolin. Path planning and robust fuzzy output-feedback control for unmanned ground vehicles with obstacle avoidance. / Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering. - (2020). - 235. 095440702097831. 10.1177/0954407020978319.

116. Using colcon to build packages. — Текст : электронный // ROS 2 Documentation : [сайт]. — URL: https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Colcon-Tutorial.html (дата обращения: 20.02.2024).

117. MLA. Summerfield, Mark. Rapid GUI Programming with Python and Qt : the Definitive Guide to PyQt Programming. Upper Saddle River, NJ :Prentice Hall, 2008.

118. Zhong, Biqing & Liu, Weihang & Zeng, Riya & Guo, Qiang & Jiang, Haibo. Autonomous Following for Unmanned Ground Vehicles on Unstructured Scenario: Risk and Performance Assessment. / Journal of Physics: Conference Series. - (2023). - 2478. 102010. 10.1088/1742-6596/2478/10/102010.

119. S, Mohanapriya & S, Mohana & T, Kumaravel & P, Sumithra. Image Detection and Segmentation using YOLO v5 for surveillance. / Applied and Computational Engineering. - (2023). - 8. 160-165. 10.54254/2755-2721/8/20230109.

120. Tran Ngoc, Hoang & Quach, Luyl Da & Nguyen, Khang & Nguyen, Huynh & Hua, Huy. Optimizing YOLO Performance for Traffic Light Detection and End-to-End Steering Control for Autonomous Vehicles in Gazebo-ROS2. /

International Journal of Advanced Computer Science and Applications. - (2023). - 14. 475-484. 10.14569/IJACSA.2023.0140752.

121. Cao, Lianjun & Zheng, Xinyu & Fang, Luming. The Semantic Segmentation of Standing Tree Images Based on the Yolo V7 Deep Learning Algorithm. / Electronics. - (2023). -12. 929. 10.3390/electronics12040929.

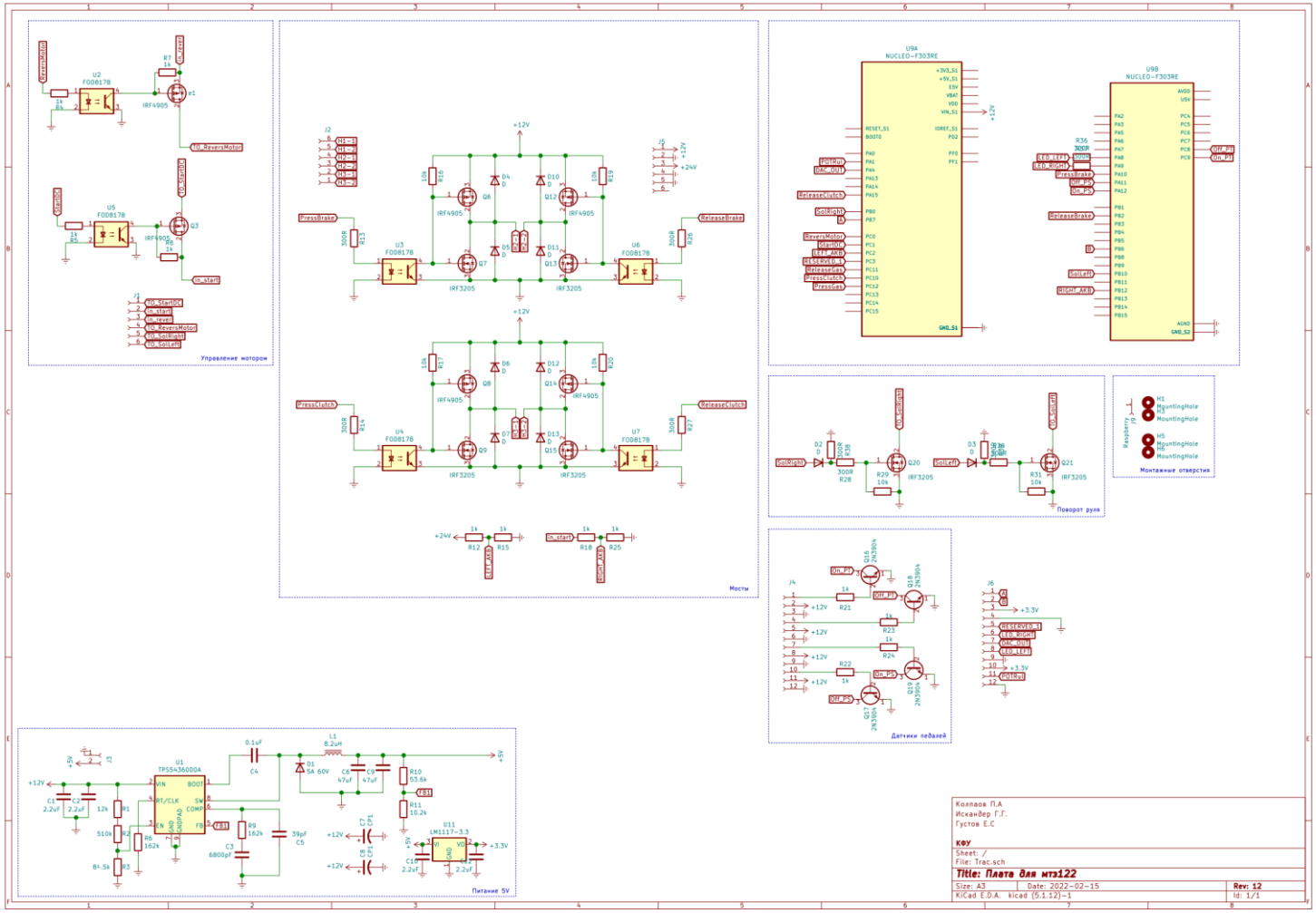
122. Barsov, Valeriy & Kosterna, Olena & Plakhotnyi, Oleksandr. RESEARCH OF THE METHODS EFFICIENCY FOR DETERMINING THE DISTANCE AND GEOMETRIC OBJECTS PARAMETERS OF TECHNICAL VISION SYSTEMS. / Advanced Information Systems. - (2020). - 4. 64-69. 10.20998/2522-9052.2020.4.09.

123. Моделирование и полунатурные испытания информационно-управляющих систем беспилотной сельскохозяйственной техники на базе трактора Беларус-3525 / И. Г. Галиуллин, Д. Е. Чикрин, Д. М. Пашин, А. Ф. Фахрутдинов // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки. – 2024. – № 5. – С. 37-43. – DOI 10.37882/2223-2966.2024.05.07. – EDN TOFBBN.

124. Моделирование системы беспилотного трактора тяжелого класса и проведение испытаний режимов управления системой на виртуальном полигоне / И. Г. Галиуллин, Д. Е. Чикрин, Д. М. Пашин, А. А. Егорчев // Имитационное моделирование. Теория и практика (ИММОД-2023) : Сборник трудов одиннадцатой всероссийской научно-практической конференции по имитационному моделированию и его применению в науке и промышленности, Казань, 18–20 октября 2023 года. – Казань: Издательство АН РТ, 2023. – С. 308-313. – EDN WUEOFK.

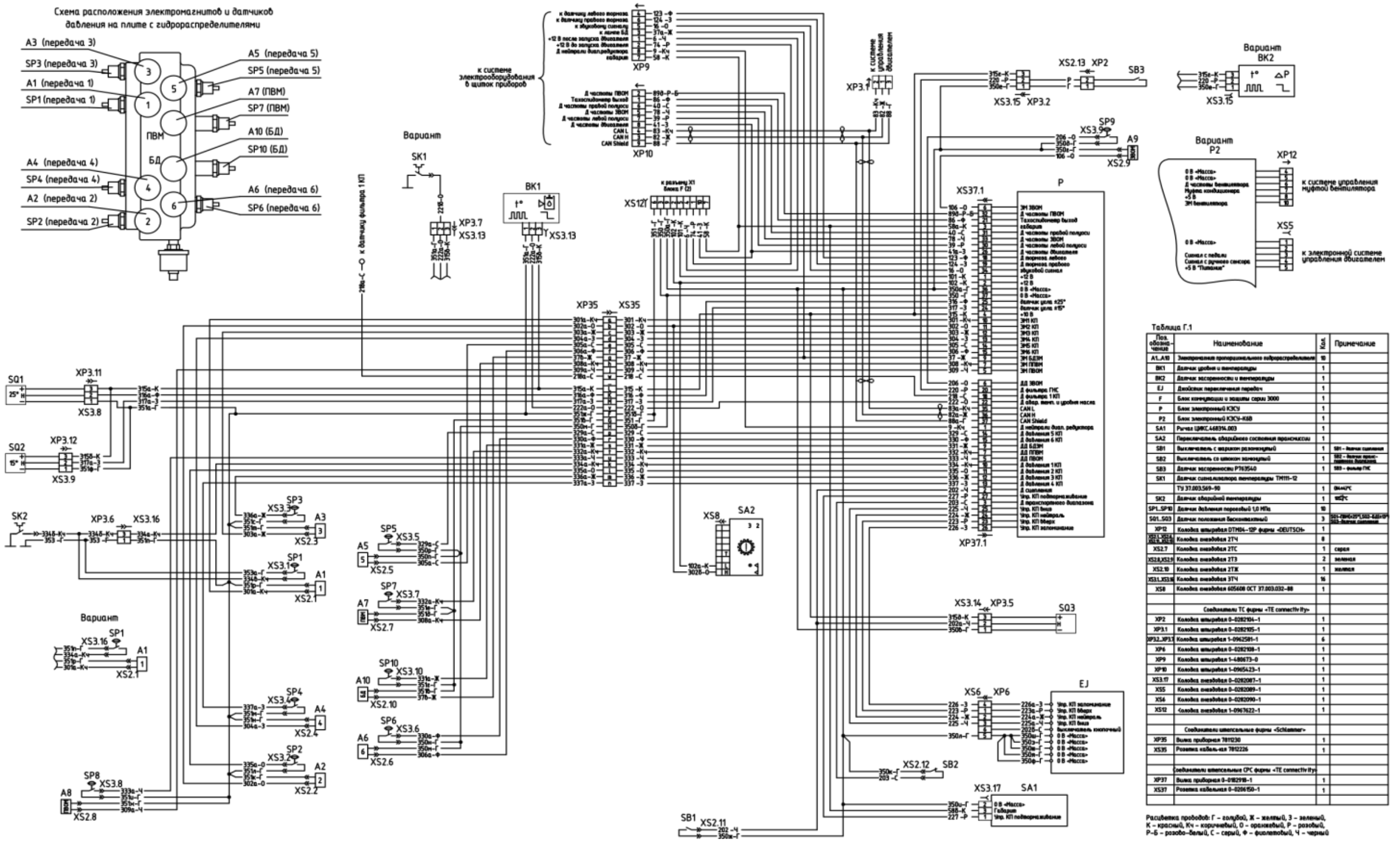
ПРИЛОЖЕНИЕ А

Принципиальная схема платы блока управления МТЗ-122.



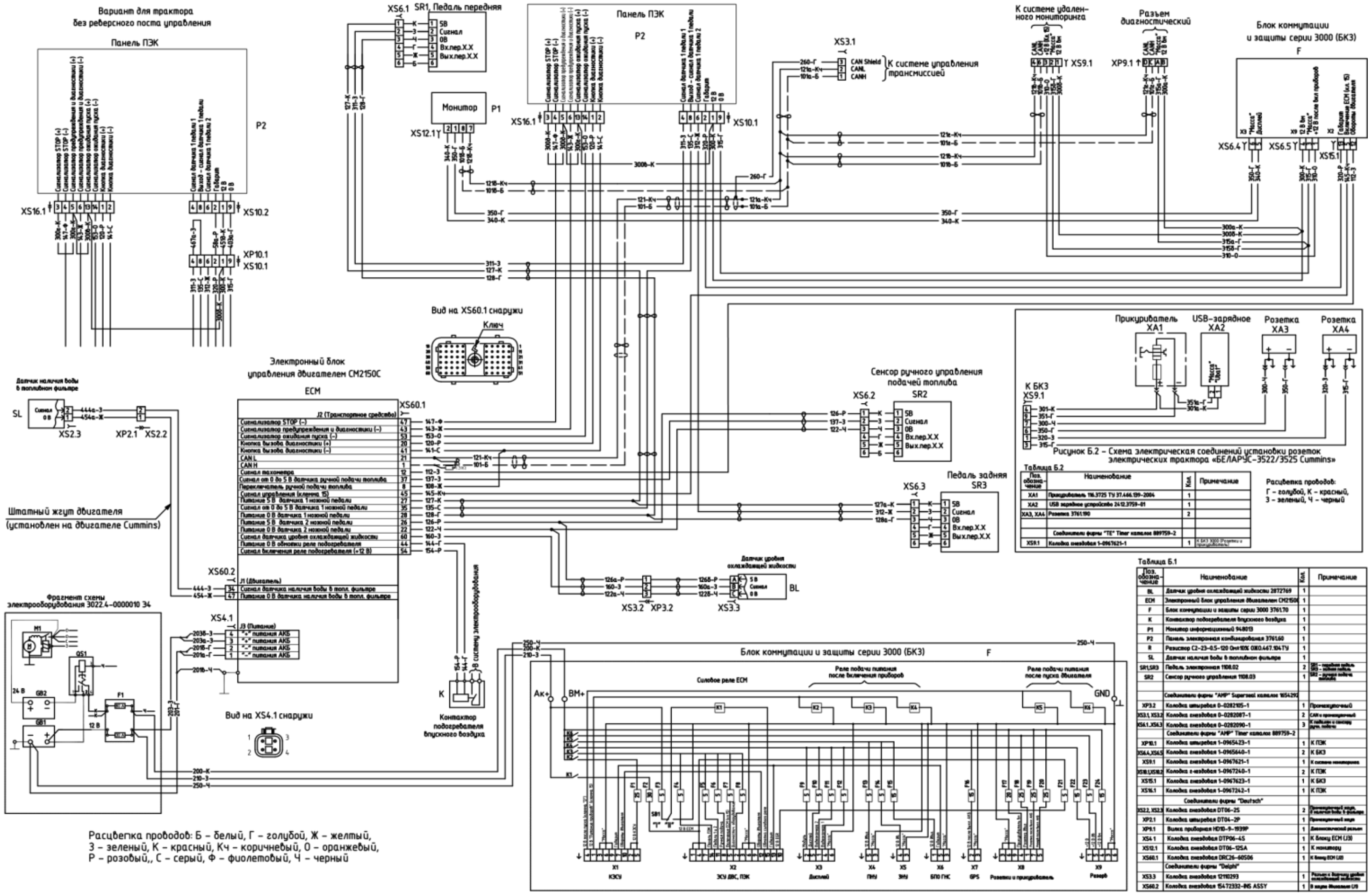
ПРИЛОЖЕНИЕ Б

Электрические соединения комплексной системы управления БД, ПВМ, ВОМ и переключением передач трактора Беларус-3522.



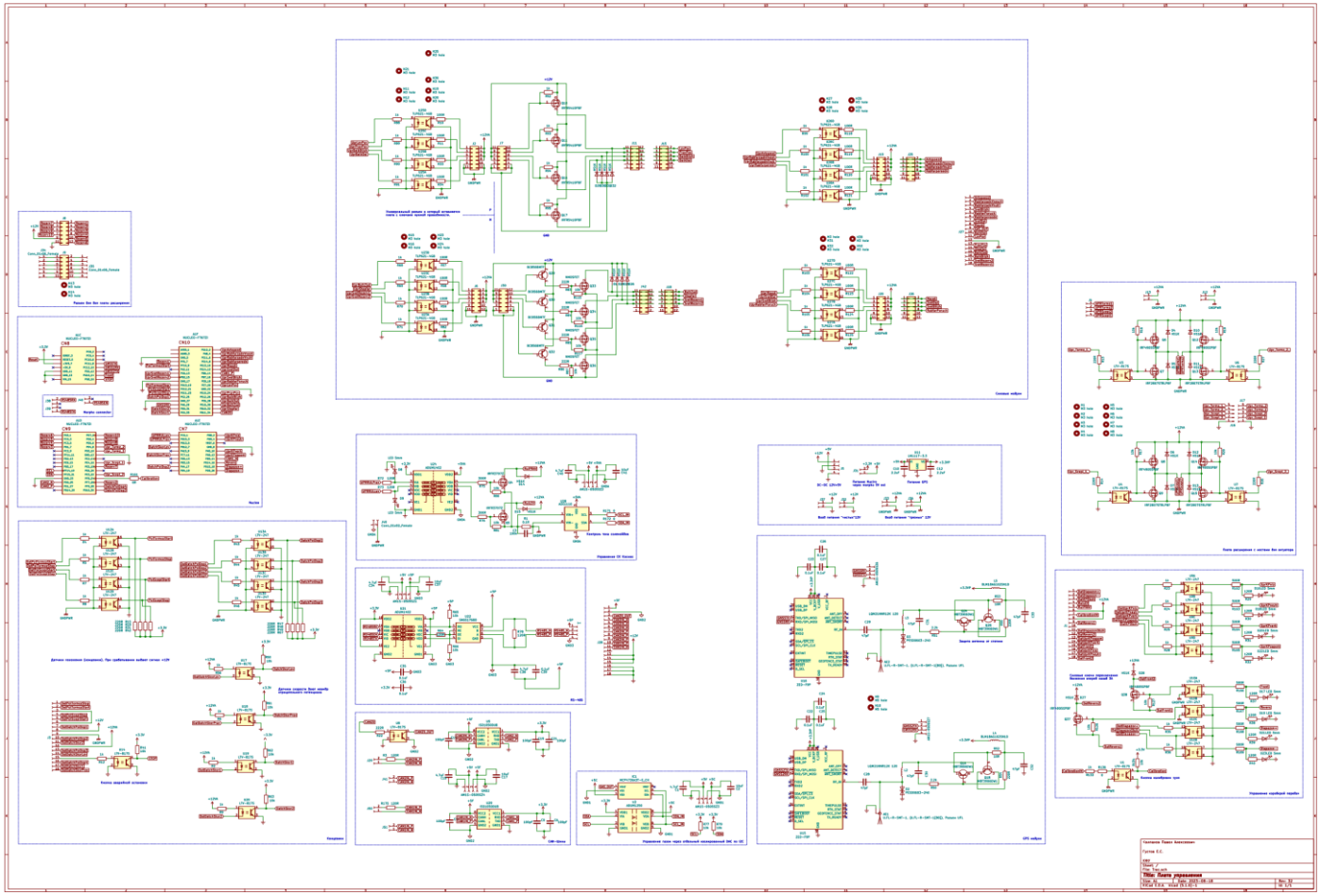
ПРИЛОЖЕНИЕ В

Электрические соединения электронной системой управления двигателем трактора Беларус-3522 Cummins.



ПРИЛОЖЕНИЕ Г

Принципиальная схема платы блока управления МТ3522.



ПРИЛОЖЕНИЕ Д

Листинг программы низкоуровневого управления системы «Беспилотный трактор КФУ-МТЗ-112»

```
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "string.h"
#include "stdlib.h"
#include "stdbool.h"
#include "stdio.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

#define STEER_RIGHT    294//1374
#define STEER_AVERAGE   574//2237
#define STEER_LEFT     855//3257

#define DAC_MAX     4095

#define DAC_TRESHOLD    0
#define timeout        1000

#define RxBufSize 1024//1024
#define alpha2 0.95

#define SS_LR          45
#define SS_RR          65
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
#define LEFT_ON() TIM_CCxChannelCmd(htim3.Instance, TIM_CHANNEL_3, TIM_CCx_ENABLE) //RIGHT
#define LEFT_OFF() TIM_CCxChannelCmd(htim3.Instance, TIM_CHANNEL_3, TIM_CCx_DISABLE) //RIGHT
#define RIGHT_ON() TIM_CCxChannelCmd(htim2.Instance, TIM_CHANNEL_3, TIM_CCx_ENABLE) //LEFT
#define RIGHT_OFF() TIM_CCxChannelCmd(htim2.Instance, TIM_CHANNEL_3, TIM_CCx_DISABLE) //LEFT
#define BUTON_OFF()  HAL_GPIO_ReadPin(BUTON_OFF_GPIO_Port, BUTON_OFF_Pin)
/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
```

```
ADC_HandleTypeDef hadc1;
ADC_HandleTypeDef hadc2;
ADC_HandleTypeDef hadc4;
DMA_HandleTypeDef hdma_adc1;

DAC_HandleTypeDef hdac1;

TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim2;
TIM_HandleTypeDef htim3;
TIM_HandleTypeDef htim7;

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */


typedef enum {
  N  = 0,
  D  = 1,
  R  = 2
} Direction_Type;

struct {
  bool autopilot_on;
  Direction_Type direction;
  int pressure;
  int steer_angle;
  int steer_angle_adc;
  bool motion;
  int manual_clutch;
  int clutch;
  bool stop;
  int acc;
  uint8_t stop_switch;
} System_state;

struct {
  int accel;
  int stop;
  int set_angle;
  Direction_Type direction;
  int Control;
}Global_state;

uint16_t acc_dac_level;
uint16_t steer_adc;

uint16_t acc_left;
uint16_t acc_right;
uint16_t acc_timer_left;
```

```
uint16_t acc_timer_right;

const char dir[3] = "NDR";
const char* autopilot[2] = {"OFF", "ON"};
const int epsilon = 10;

uint8_t receiveBuffer[RxBufSize];
char sendBuffer[RxBufSize];
char sendBuffer2[RxBufSize];

char *pars[12];
uint16_t tickstart;
bool command_received = false;
bool direction_changed = false;

uint32_t counter_toggle=0;
uint16_t adc4Result=0;
uint16_t procent_epwm=0;
uint32_t counter_adc=1;
uint32_t sum_adc=0;
int avr_steer_angle=0;
int filter_steer_angle=0;
int8_t test_dir=0;
uint8_t flag_off_tumbler=0;
uint8_t l=0;
uint8_t counter_com_tx=0;
struct {
  uint8_t Press_Brake_State;
  uint8_t Release_Brake_State;
  uint8_t Sensor_Press_Brake;
  uint8_t Sensor_Release_Brake;

  uint8_t Press_Gas_State;
  uint8_t Release_Gas_State;

  uint8_t Press_Clutch_State;
  uint8_t Release_Clutch_State;
  uint8_t Sensor_Press_Clutch;
  uint8_t Sensor_Release_Clutch;

  uint16_t Press_Brake_counter;
  uint16_t Release_Brake_counter;
  uint8_t Brake_Error;

  uint16_t Press_Gas_counter;
  uint16_t Release_Gas_counter;
  uint8_t Gas_Error;

  uint16_t Press_Clutch_counter;
  uint16_t Release_Clutch_counter;
  uint8_t Clutch_Error;
```

```
}Sensor_state={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

struct comand_pedals{
uint8_t comand_press_brake;
uint8_t comand_release_brake;
uint8_t comand_press_gas;
uint8_t comand_release_gas;
uint8_t comand_press_clutch;
uint8_t comand_release_clutch;
}Comand_pedals={0,0,0,0,0,0};

struct encoder_gas_state{
  uint32_t counter_raw;
  uint8_t direction;
}Encoder_gas_state={0,0};
uint8_t Buton_off=1;
uint16_t Odometr_left_timer=0;
uint16_t Odometr_right_timer=0;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_ADC1_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_DAC1_Init(void);
static void MX_TIM1_Init(void);
static void MX_TIM2_Init(void);
static void MX_TIM3_Init(void);
static void MX_TIM7_Init(void);
static void MX_ADC2_Init(void);
static void MX_ADC4_Init(void);
/* USER CODE BEGIN PFP */
void set_steer_angle(void);
int get_steer_angle(void);
void set_acceleration(uint8_t level);
void start_motion(uint8_t direction);
void stop_motion(void);
void turn_off_autopilot(void);
void error_brake(void);
void press_brake(void);
void release_bake(void);
void press_clutch(uint8_t press);
void led_adc_set(uint8_t left_adc_gpio,uint8_t right_adc_gpio);
void led_adc_timer(void);

extern void TX_UART_f(void);
uint8_t press_command=0;
extern void Odometr_function(uint16_t timer_left, uint16_t timer_right);
```

```
extern void Panel_functions(uint16_t timer_left, uint16_t timer_right, uint16_t acc_left, uint16_t
acc_right);
extern uint8_t buton_funciton(uint8_t gpio_state);
void braking(void);
/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
extern uint8_t flag_Timer7;
extern uint16_t Filter_SMA(uint16_t For_Filtered);
extern uint16_t Filter_Buffer[FILTER_SMA_ORDER];
extern uint8_t tx_buff[47];
extern tx_uart_my TX_Uart_my;
extern odometr_struct Odometr_left;
extern odometr_struct Odometr_right;
extern panel_struct Panel_struct;
uint8_t Pars_String(char **pars, char* str)
{
  char *pch;
  int y = 0;
  pch = strtok (str,"\",");
  while (pch != NULL)
    {
      pars[y++]=pch;
      pch = strtok (NULL, "\",");
    }
  return y;
}

void CheckUart()
{
  uint8_t *start_s_nump;
            uint8_t *start_s_nump_t;
  uint8_t start_s_num = 0;
#ifndef TEST
//  if(command_received)
//    if((HAL_GetTick() - tickstart) > timeout)
//      Global_state.stop = 1;
#endif

            /* P—P°C‰PëC,P° PsC, PïPµCЂPµPïPsP»PSPµPSPëC? P±CґC„PµCЂPsPI */
  if (huart2.RxXferCount <=0x80 )
                      {
                      /* PћC‡PëC?C,PєP° P±CґC„PµCЂPsPI */
     memset(receiveBuffer, 0, RxBufSize);
     HAL_UART_AbortReceive_IT(&huart2);
     HAL_UART_Receive_IT(&huart2, receiveBuffer, RxBufSize);
                      }

  /* P•C?P»Pë PSP°PNPrPµPS C?PëPjPIPsP» PєPsPSC†P° PïP°PєPµC,P° */
  if (strchr((char*)receiveBuffer, '#') != NULL )
```

```
  {

    /* P•C?P»Pë PSP°PNºPrPµPS C?PëPjPIPsP» PSP°C‡P°P»P° PïP°PєPµC,P° */
                        /* PïC‰PµPj PïPsC?P»PµPrPSPëPNº C?PëPjPIPsP» PSP°C‡P°P»P° PïP°PєPµC,P° PI
PïPsC?C‹P»PєPµ */
                        start_s_nump_t = receiveBuffer;
                        start_s_nump = NULL;
                        do
                                {
                                        start_s_nump_t = (uint8_t*)strchr((char*)start_s_nump_t, '*');
                                        if ( (start_s_nump_t !=NULL)  && (strchr((char*)start_s_nump_t,
'#' ) != NULL) )
                                                {
                                                        start_s_nump = start_s_nump_t;
                                                        start_s_nump_t++;
                                                }
                                } while (start_s_nump_t != NULL);

                        if ( start_s_nump == NULL)
                                return;
                    //  TX_UART_f();
                //   memset(sendBuffer2, 0, 128);
                        //memcpy(sendBuffer2, receiveBuffer, strlen(receiveBuffer));
                        //HAL_UART_Transmit_IT(&huart2, sendBuffer2, strlen(sendBuffer2));
                        //memset(sendBuffer2, 0, 128);
                        //memcpy(sendBuffer2, tx_buff, (strlen(tx_buff)+1));
                        //HAL_UART_Transmit_IT(&huart2, (uint8_t*)tx_buff, 42);


    if (start_s_nump != NULL)
     {
      start_s_num = start_s_nump - receiveBuffer;
      Pars_String(pars, (char*)start_s_nump + 1);
      //checkXOR();

      Global_state.accel = atoi(pars[0]);

      if (pars[1][0] == '1')
        Global_state.stop = 1;
      else
        Global_state.stop = 0;

      Global_state.set_angle = atoi(pars[2]);

     switch (pars[3][0])
      {
        case 'N':
          Global_state.direction = N;
          break;
        case 'D':
          {
```

```
            Global_state.direction = R;
           // TX_UART_f();
           // HAL_UART_Transmit_IT(&huart2, (uint8_t*)tx_buff, 42);
           break;
          }
        case 'R':
          Global_state.direction = D;
          break;
        default:
          Global_state.direction = N;
          break;
       }
       if ( !strcmp("ON", pars[4]) )
         Global_state.Control = 1;
       else
         Global_state.Control = 0;

       tickstart = HAL_GetTick();
       command_received = true;
      }

    /* PћC‡PëC?C,PєP° P±CґC„PµCЂPsPl */
    memset(receiveBuffer, 0, RxBufSize);
    HAL_UART_AbortReceive_IT(&huart2);
    HAL_UART_Receive_IT(&huart2, receiveBuffer, RxBufSize);
   }
}
void epvm_potenciometr(void)
{
//  HAL_ADC_Start(&hadc4);
                                                  //HAL_ADCEx_InjectedPollForConversion(&hadc4, 10);
 // adc4Result = HAL_ADC_GetValue(&hadc4);

  // TIM2->CCR3=(100*adc4Result)/4033;
  procent_epwm=TIM2->CCR3*100/100;
 // HAL_ADC_Stop(&hadc4);

   //HAL_Delay(300);
}
void avr_potenciometr(int row_angle, uint16_t max_counter_adc)
{
  counter_adc++;
    if( counter_adc>=max_counter_adc)
    {
      avr_steer_angle=sum_adc/counter_adc;
      counter_adc=0;
      sum_adc=row_angle;
    }
    else
    {
      sum_adc=sum_adc+row_angle;
```

```
   }
}
void Encoder_gas(void)
{
 Encoder_gas_state.counter_raw=TIM4->CNT;
 Encoder_gas_state.direction=((TIM4->CR1)&0x00000010)>1;
 /*TX_Uart_my.gas_encoder_state[0]=Encoder_gas_state.counter_raw&0x000F;
 TX_Uart_my.gas_encoder_state[1]=Encoder_gas_state.counter_raw&0x00F0>>8;
 TX_Uart_my.gas_encoder_state[2]=Encoder_gas_state.counter_raw&0x0F00>>16;
 TX_Uart_my.gas_encoder_state[3]=Encoder_gas_state.counter_raw&0xF000>>24; */
}
void Sensors_Condition (void)
{
//read comand gpio on Brake
 Sensor_state.Press_Brake_State=HAL_GPIO_ReadPin(GPIOA,PRESS_BRAKE_Pin);
 Sensor_state.Release_Brake_State=HAL_GPIO_ReadPin(GPIOB,RELEASE_BRAKE_Pin);
//read sensror gpio on Brake
 Sensor_state.Sensor_Press_Brake=HAL_GPIO_ReadPin(GPIOC,SENSOR_PRESS_BRAKE_Pin);
 Sensor_state.Sensor_Release_Brake=HAL_GPIO_ReadPin(GPIOC,SENSOR_RELEASE_BRAKE_Pin);
//read comand gpio on Clutch
 Sensor_state.Press_Clutch_State=HAL_GPIO_ReadPin(GPIOC,PRESS_CLUTCH_Pin);
 Sensor_state.Release_Clutch_State=HAL_GPIO_ReadPin(GPIOA,RELEASE_CLUTCH_Pin);

 //read sensor gpio on Clutch
 Sensor_state.Sensor_Press_Clutch=HAL_GPIO_ReadPin(GPIOA,SENSOR_PRESS_CLUTCH_Pin);
 Odometr_right.new_state_sensor=Sensor_state.Sensor_Press_Clutch;

 Sensor_state.Sensor_Release_Clutch=HAL_GPIO_ReadPin(GPIOA,SENSOR_RELEASE_CLUTCH_Pin);
 Odometr_left.new_state_sensor=Sensor_state.Sensor_Release_Clutch;

//read comand gpio on Gas
 Sensor_state.Press_Gas_State=HAL_GPIO_ReadPin(GPIOC,PRESS_GAS_Pin);
 Sensor_state.Release_Gas_State=HAL_GPIO_ReadPin(GPIOC,RELEASE_GAS_Pin);

 TX_Uart_my.brake_sensor_state=Sensor_state.Sensor_Press_Brake*10+
Sensor_state.Sensor_Release_Brake;
 // TX_Uart_my.clutch_sensor_state=Sensor_state.Sensor_Press_Clutch*10+
Sensor_state.Sensor_Release_Clutch;

 led_adc_set(Panel_struct.write_gpio_left,Panel_struct.write_gpio_right);
 //Encoder_gas();
}
void error_brake(void)
{
 if((Sensor_state.Press_Brake_State==1)&(Sensor_state.Release_Brake_State==1))
 {

   HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
   HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
   Sensor_state.Brake_Error=1;
   for(uint8_t i=0; i>=250;i++){}
```

```
  }
  else
  {
   if((Sensor_state.Sensor_Press_Brake==0)&(Sensor_state.Sensor_Release_Brake==0))
    {
     HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
     HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
     Sensor_state.Brake_Error=2;
     TX_Uart_my.brake_state=2;
     for(uint8_t i=0; i>=250;i++){}
    }
    else{
     if(Sensor_state.Sensor_Press_Brake==0){TX_Uart_my.brake_state=1;}
     if(Sensor_state.Sensor_Release_Brake==0){TX_Uart_my.brake_state=0;}
     Sensor_state.Brake_Error=0;}
  }
if((Sensor_state.Press_Clutch_State==1)&(Sensor_state.Release_Clutch_State==1))
  {
   HAL_GPIO_WritePin(GPIOC,PRESS_CLUTCH_Pin,GPIO_PIN_RESET);
   HAL_GPIO_WritePin(GPIOA,RELEASE_CLUTCH_Pin,GPIO_PIN_RESET);
   Sensor_state.Brake_Error=1;
   for(uint8_t i=0; i>=250;i++){}
  }
  else
  {
   if((Sensor_state.Sensor_Press_Clutch==0)&(Sensor_state.Sensor_Release_Clutch==0))
    {
     HAL_GPIO_WritePin(GPIOC,PRESS_CLUTCH_Pin,GPIO_PIN_RESET);
     HAL_GPIO_WritePin(GPIOA,RELEASE_CLUTCH_Pin,GPIO_PIN_RESET);
     Sensor_state.Clutch_Error=2;
     for(uint8_t i=0; i>=250;i++){}
    }
    else{Sensor_state.Clutch_Error=0;}
  }
}

void press_brake(void){

    if(Sensor_state.Sensor_Press_Brake==0)
    {
     HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
     HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
     for(uint8_t i=0; i>=250;i++){}
    }//sensor lights on
    else
    {

     if(Sensor_state.Brake_Error==0)
     {
       HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_SET);
       HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
```

```
        for(uint8_t i=0; i>=250;i++){}
      }
      else
      {
       HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
       HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
      }
    }

}
void release_bake(void){

   if(Sensor_state.Sensor_Release_Brake==0)
   {
     HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
     HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
     for(uint8_t i=0; i>=250;i++){}
   }//sensor lights on
   else
   {
    if(Sensor_state.Brake_Error==0)
    {
     HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
     HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_SET);
     for(uint8_t i=0; i>=250;i++){}
    }
    else
    {
      HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
      HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
    }
   }

}
void press_clutch(uint8_t press)
{
  switch (press)
   {
   case 0:
    {

     HAL_GPIO_WritePin(GPIOC,PRESS_CLUTCH_Pin,GPIO_PIN_RESET);
     HAL_GPIO_WritePin(GPIOA,RELEASE_CLUTCH_Pin,GPIO_PIN_RESET);
     break;
    }
   case 1:
    {

     if(Sensor_state.Sensor_Press_Clutch==1)
     {
      HAL_GPIO_WritePin(GPIOC,PRESS_CLUTCH_Pin,GPIO_PIN_SET);
```

```
        HAL_GPIO_WritePin(GPIOA,RELEASE_CLUTCH_Pin,GPIO_PIN_RESET);
       }
       else
       {
        HAL_GPIO_WritePin(GPIOC,PRESS_CLUTCH_Pin,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,RELEASE_CLUTCH_Pin,GPIO_PIN_RESET);
       }
       break;
     }
    case 2:
     {
       if(Sensor_state.Sensor_Release_Clutch==1)
       {
        HAL_GPIO_WritePin(GPIOC,PRESS_CLUTCH_Pin,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,RELEASE_CLUTCH_Pin,GPIO_PIN_SET);
       }
       else
       {
        HAL_GPIO_WritePin(GPIOC,PRESS_CLUTCH_Pin,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,RELEASE_CLUTCH_Pin,GPIO_PIN_RESET);
       }
       break;
     }
    }
}

void Comands_Pedals_Condition(void)
{
//  Brake


HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
    Sensor_state.Brake_Error=10;
//Clutch
}
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

/*Button_state.last_state=1;
Button_state.new_state=1;
Button_state.counter_flag_state=0;
Button_state.flag_Start_PWM=0;*/
  /* USER CODE END 1 */
```

```
 /* MCU Configuration--------------------------------------------------------*/

 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
 HAL_Init();

 /* USER CODE BEGIN Init */

 /* USER CODE END Init */

 /* Configure the system clock */
 SystemClock_Config();

 /* USER CODE BEGIN SysInit */

 /* USER CODE END SysInit */

 /* Initialize all configured peripherals */
 MX_GPIO_Init();
 MX_DMA_Init();
 MX_ADC1_Init();
 MX_USART2_UART_Init();
 MX_DAC1_Init();
 MX_TIM1_Init();
 MX_TIM2_Init();
 MX_TIM3_Init();
 MX_TIM7_Init();
 MX_ADC2_Init();
 MX_ADC4_Init();
 /* USER CODE BEGIN 2 */
 HAL_UART_Receive_IT(&huart2,          receiveBuffer, RxBufSize);

 //HAL_TIM_Base_Start(&htim1);
 HAL_TIM_Base_Start(&htim2);
 HAL_TIM_Base_Start(&htim3);



 HAL_DAC_Start(&hdac1, DAC_CHANNEL_1);

 set_acceleration(25);

//  set_acceleration(70);
//
//  set_acceleration(100);
 //HAL_TIM_PWM_Start (&htim1, TIM_CHANNEL_1);
HAL_TIM_PWM_Start (&htim3, TIM_CHANNEL_3);
 //HAL_TIM_PWM_Start_IT (&htim3, TIM_CHANNEL_3);
//HAL_TIM_Encoder_Start (&htim4, TIM_CHANNEL_1|TIM_CHANNEL_2);
HAL_TIM_Base_Start_IT(&htim7);
 HAL_ADC_Start_DMA(&hadc1, (uint32_t *)&steer_adc, 1);
```

```
  HAL_ADC_Start(&hadc1);
  //HAL_ADC_Start_DMA(&hadc2, (uint32_t *)&acc_left, 1);
  HAL_ADC_Start(&hadc2);
  //HAL_ADC_Start_DMA(&hadc4, (uint32_t *)&acc_right, 1);
  HAL_ADC_Start(&hadc4);
//  TIM1->CCR1=SS_LR;
  TIM2->CCR3=SS_LR;
  TIM3->CCR3=SS_RR;//right

LEFT_OFF();
RIGHT_OFF();
 memset(receiveBuffer, 0, 1024);
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
            System_state.steer_angle_adc= avr_steer_angle;
            System_state.steer_angle_adc = (uint16_t)((float)System_state.steer_angle_adc*alpha2 +
(float)steer_adc*(1-alpha2));
            System_state.steer_angle = get_steer_angle();
            CheckUart();
            Buton_off=buton_funciton(BUTON_OFF());

             if(Buton_off==0)
             {
               if(Global_state.Control)
                {
                                                  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
                    //HAL_Delay(300);
                    if(Global_state.stop == 1)
                    {
                      stop_motion();
                      if(Global_state.set_angle != System_state.steer_angle){set_steer_angle();}
                    }
                    else
                    {
                      start_motion(Global_state.direction);
                      if(Global_state.set_angle != System_state.steer_angle){set_steer_angle();}
                    }
                 }
                 else
                 {
                   HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
                   turn_off_autopilot();
                 }
             }
             else{
              //Buton_off=1;
```

```
            // System_state.acc==0;
             stop_motion();
             RIGHT_OFF(); //RIGHT
             LEFT_OFF();
            }

            error_brake();
            filter_steer_angle=Filter_SMA(steer_adc);
            avr_steer_angle= filter_steer_angle;
            Sensors_Condition();
            Panel_functions(acc_timer_left, acc_timer_right, acc_left, acc_right);

            TX_Uart_my.motion_state=Global_state.Control;
            TX_Uart_my.DAC_electric_motor[0]=System_state.acc;

          //
if(System_state.acc==0){Odometr_left.angular_velociy[0]=0;Odometr_right.angular_velociy[0]=0;}
          //  else
           /* {
                //if(test_dir>0)
                if(HAL_GPIO_ReadPin(REVERSE_GPIO_Port, REVERSE_Pin)>0)
                {
                   Odometr_left.new_state_direciton=1;
                   Odometr_right.new_state_direciton=1;
                }
                else
                {
                   Odometr_left.new_state_direciton=2;
                   Odometr_right.new_state_direciton=2;
                }*/
                                          Odometr_function(Odometr_left_timer,
Odometr_right_timer);
           // }
            TX_Uart_my.odometr_left[0]=Odometr_left.angular_velociy[0]&0x000000FF;
            TX_Uart_my.odometr_left[1]=(Odometr_left.angular_velociy[0]&0x0000FF00)>>8;
            TX_Uart_my.odometr_left[2]=(Odometr_left.angular_velociy[0]&0x00FF0000)>>16;
            TX_Uart_my.odometr_left[3]=(Odometr_left.angular_velociy[0]&0xFF000000)>>24;

            TX_Uart_my.odometr_right[0]=Odometr_right.angular_velociy[0]&0x000000FF;
            TX_Uart_my.odometr_right[1]=(Odometr_right.angular_velociy[0]&0x0000FF00)>>8;
            TX_Uart_my.odometr_right[2]=(Odometr_right.angular_velociy[0]&0x00FF0000)>>16;
            TX_Uart_my.odometr_right[3]=(Odometr_right.angular_velociy[0]&0xFF000000)>>24;

            TX_Uart_my.DAC_electric_motor[1]=0;
            TX_Uart_my.DAC_electric_motor[2]=0;

            TX_Uart_my.U_wheel_angle[0]=filter_steer_angle&0x00FF;
            TX_Uart_my.U_wheel_angle[1]=(filter_steer_angle&0xFF00)>>8;
            TX_Uart_my.U_wheel_angle[2]=0;
            TX_Uart_my.U_wheel_angle[3]=0;
```

```
            TX_Uart_my.wheel_angle[0]= System_state.steer_angle;
            TX_Uart_my.wheel_angle[1]=0;
            TX_Uart_my.wheel_angle[2]=0;
            TX_Uart_my.wheel_angle[3]=0;


                switch(System_state.direction)
               {
                case N:
                {
                  TX_Uart_my.control_condition[0]='N';
                  break;
                }
                case R:
                {
                  TX_Uart_my.control_condition[0]='D';
                  break;
                }case D:
                {
                  TX_Uart_my.control_condition[0]='R';
                  break;
                }
               }


//if(System_state.acc==0){Odometr_left.angular_velociy[0]=0;Odometr_right.angular_velociy[0]=0;}
/*if(test_dir==0){}
else
{
    //if(test_dir>0)
                                                 if(HAL_GPIO_ReadPin(REVERSE_GPIO_Port, REVERSE_Pin)>0)
    {
       Odometr_left.new_state_direciton=1;
       Odometr_right.new_state_direciton=1;
    }
    else
    {

       Odometr_left.new_state_direciton=2;
       Odometr_right.new_state_direciton=2;
    }

}*/
Odometr_function(Odometr_left_timer, Odometr_right_timer);
// l++;
//  if(Odometr_left.angular_velociy[0]==0){Odometr_left.angular_velociy[0]=0;}
// if(Odometr_right.angular_velociy[0]==0){Odometr_right.angular_velociy[0]=112+l;}
// if(l>=255){l=0;}

TX_Uart_my.odometr_left[0]=Odometr_left.angular_velociy[0]&0x000000FF;
TX_Uart_my.odometr_left[1]=(Odometr_left.angular_velociy[0]&0x0000FF00)>>8;
```

```
  TX_Uart_my.odometr_left[2]=(Odometr_left.angular_velociy[0]&0x00FF0000)>>16;
  TX_Uart_my.odometr_left[3]=(Odometr_left.angular_velociy[0]&0xFF000000)>>24;

  TX_Uart_my.odometr_right[0]=Odometr_right.angular_velociy[0]&0x000000FF;
  TX_Uart_my.odometr_right[1]=(Odometr_right.angular_velociy[0]&0x0000FF00)>>8;
  TX_Uart_my.odometr_right[2]=(Odometr_right.angular_velociy[0]&0x00FF0000)>>16;
  TX_Uart_my.odometr_right[3]=(Odometr_right.angular_velociy[0]&0xFF000000)>>24;

  TX_Uart_my.DAC_electric_motor[1]=0;
  TX_Uart_my.DAC_electric_motor[2]=0;

  TX_Uart_my.U_wheel_angle[0]=filter_steer_angle&0x00FF;
    TX_Uart_my.U_wheel_angle[1]=(filter_steer_angle&0xFF00)>>8;
     TX_Uart_my.U_wheel_angle[2]=0;
      TX_Uart_my.U_wheel_angle[3]=0;

  TX_Uart_my.wheel_angle[0]= System_state.steer_angle;
   TX_Uart_my.wheel_angle[1]=0;
    TX_Uart_my.wheel_angle[2]=0;
     TX_Uart_my.wheel_angle[3]=0;


      switch(System_state.direction)
     {
     case N:
      {
        TX_Uart_my.control_condition[0]='N';
        break;
      }
      case R:
      {
        TX_Uart_my.control_condition[0]='D';
        break;
      }case D:
      {
        TX_Uart_my.control_condition[0]='R';
        break;
      }
     }


  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
```

```
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  RCC_OscInitStruct.PLL.PREDIV = RCC_PREDIV_DIV1;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART2|RCC_PERIPHCLK_TIM1
                              |RCC_PERIPHCLK_ADC12|RCC_PERIPHCLK_ADC34
                              |RCC_PERIPHCLK_TIM2|RCC_PERIPHCLK_TIM34;
  PeriphClkInit.Usart2ClockSelection = RCC_USART2CLKSOURCE_PCLK1;
  PeriphClkInit.Adc12ClockSelection = RCC_ADC12PLLCLK_DIV6;
  PeriphClkInit.Adc34ClockSelection = RCC_ADC34PLLCLK_DIV6;
  PeriphClkInit.Tim1ClockSelection = RCC_TIM1CLK_HCLK;
  PeriphClkInit.Tim2ClockSelection = RCC_TIM2CLK_HCLK;
  PeriphClkInit.Tim34ClockSelection = RCC_TIM34CLK_HCLK;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief ADC1 Initialization Function
  * @param None
  * @retval None
  */
```

```
static void MX_ADC1_Init(void)
{

  /* USER CODE BEGIN ADC1_Init 0 */

  /* USER CODE END ADC1_Init 0 */

  ADC_MultiModeTypeDef multimode = {0};
  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC1_Init 1 */

  /* USER CODE END ADC1_Init 1 */
  /** Common config
  */
  hadc1.Instance = ADC1;
  hadc1.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
  hadc1.Init.Resolution = ADC_RESOLUTION_10B;
  hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc1.Init.ContinuousConvMode = ENABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_RISING;
  hadc1.Init.ExternalTrigConv = ADC_EXTERNALTRIGCONV_T3_TRGO;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  hadc1.Init.DMAContinuousRequests = ENABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  hadc1.Init.LowPowerAutoWait = DISABLE;
  hadc1.Init.Overrun = ADC_OVR_DATA_OVERWRITTEN;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure the ADC multi-mode
  */
  multimode.Mode = ADC_MODE_INDEPENDENT;
  if (HAL_ADCEx_MultiModeConfigChannel(&hadc1, &multimode) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure Regular Channel
  */
  sConfig.Channel = ADC_CHANNEL_2;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SingleDiff = ADC_SINGLE_ENDED;
  sConfig.SamplingTime = ADC_SAMPLETIME_181CYCLES_5;
  sConfig.OffsetNumber = ADC_OFFSET_NONE;
  sConfig.Offset = 0;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
```

```
  }
  /* USER CODE BEGIN ADC1_Init 2 */

  /* USER CODE END ADC1_Init 2 */

}

/**
  * @brief ADC2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC2_Init(void)
{

  /* USER CODE BEGIN ADC2_Init 0 */

  /* USER CODE END ADC2_Init 0 */

  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC2_Init 1 */

  /* USER CODE END ADC2_Init 1 */
  /** Common config
  */
  hadc2.Instance = ADC2;
  hadc2.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
  hadc2.Init.Resolution = ADC_RESOLUTION_12B;
  hadc2.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc2.Init.ContinuousConvMode =  ENABLE;
  hadc2.Init.DiscontinuousConvMode = DISABLE;
  hadc2.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_RISING;
  hadc2.Init.ExternalTrigConv = ADC_EXTERNALTRIGCONV_T3_TRGO;
  hadc2.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc2.Init.NbrOfConversion = 1;
  hadc2.Init.DMAContinuousRequests = DISABLE;
  hadc2.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  hadc2.Init.LowPowerAutoWait = DISABLE;
  hadc2.Init.Overrun = ADC_OVR_DATA_OVERWRITTEN;
  if (HAL_ADC_Init(&hadc2) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure Regular Channel
  */
  sConfig.Channel = ADC_CHANNEL_8;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SingleDiff = ADC_SINGLE_ENDED;
  sConfig.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
  sConfig.OffsetNumber = ADC_OFFSET_NONE;
```

```
  sConfig.Offset = 0;
  if (HAL_ADC_ConfigChannel(&hadc2, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN ADC2_Init 2 */

  /* USER CODE END ADC2_Init 2 */

}

/**
  * @brief ADC4 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC4_Init(void)
{

  /* USER CODE BEGIN ADC4_Init 0 */

  /* USER CODE END ADC4_Init 0 */

  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC4_Init 1 */

  /* USER CODE END ADC4_Init 1 */
  /** Common config
  */
  hadc4.Instance = ADC4;
  hadc4.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
  hadc4.Init.Resolution = ADC_RESOLUTION_12B;
  hadc4.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc4.Init.ContinuousConvMode = ENABLE;
  hadc4.Init.DiscontinuousConvMode = DISABLE;
  hadc4.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc4.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc4.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc4.Init.NbrOfConversion = 1;
  hadc4.Init.DMAContinuousRequests = DISABLE;
  hadc4.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  hadc4.Init.LowPowerAutoWait = DISABLE;
  hadc4.Init.Overrun = ADC_OVR_DATA_OVERWRITTEN;
  if (HAL_ADC_Init(&hadc4) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure Regular Channel
  */
  sConfig.Channel = ADC_CHANNEL_3;
```

```
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SingleDiff = ADC_SINGLE_ENDED;
  sConfig.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
  sConfig.OffsetNumber = ADC_OFFSET_NONE;
  sConfig.Offset = 0;
  if (HAL_ADC_ConfigChannel(&hadc4, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN ADC4_Init 2 */

  /* USER CODE END ADC4_Init 2 */

}

/**
  * @brief DAC1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_DAC1_Init(void)
{

  /* USER CODE BEGIN DAC1_Init 0 */

  /* USER CODE END DAC1_Init 0 */

  DAC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN DAC1_Init 1 */

  /* USER CODE END DAC1_Init 1 */
  /** DAC Initialization
  */
  hdac1.Instance = DAC1;
  if (HAL_DAC_Init(&hdac1) != HAL_OK)
  {
    Error_Handler();
  }
  /** DAC channel OUT1 config
  */
  sConfig.DAC_Trigger = DAC_TRIGGER_NONE;
  sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;
  if (HAL_DAC_ConfigChannel(&hdac1, &sConfig, DAC_CHANNEL_1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN DAC1_Init 2 */

  /* USER CODE END DAC1_Init 2 */
```

```
}

/**
  * @brief TIM1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM1_Init(void)
{

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 720-1;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 100-1;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterOutputTrigger2 = TIM_TRGO2_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */

  /* USER CODE END TIM1_Init 2 */

}

/**
  * @brief TIM2 Initialization Function
```

```
  * @param None
  * @retval None
  */
static void MX_TIM2_Init(void)
{

  /* USER CODE BEGIN TIM2_Init 0 */

  /* USER CODE END TIM2_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM2_Init 1 */

  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 720-1;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 100-1;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 26;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_3) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */
```

```
  /* USER CODE END TIM2_Init 2 */
  HAL_TIM_MspPostInit(&htim2);

}

/**
  * @brief TIM3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM3_Init(void)
{

  /* USER CODE BEGIN TIM3_Init 0 */

  /* USER CODE END TIM3_Init 0 */

  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM3_Init 1 */

  /* USER CODE END TIM3_Init 1 */
  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 720-1;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim3.Init.Period = 100-1;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_PWM_Init(&htim3) !=  HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_3) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */
```

```
  HAL_TIM_MspPostInit(&htim3);

}

/**
  * @brief TIM7 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM7_Init(void)
{

  /* USER CODE BEGIN TIM7_Init 0 */

  /* USER CODE END TIM7_Init 0 */

  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM7_Init 1 */

  /* USER CODE END TIM7_Init 1 */
  htim7.Instance = TIM7;
  htim7.Init.Prescaler = 1800;
  htim7.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim7.Init.Period = 100;
  htim7.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim7) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim7, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM7_Init 2 */

  /* USER CODE END TIM7_Init 2 */

}

/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{

  /* USER CODE BEGIN USART2_Init 0 */
```

```
  /* USER CODE END USART2_Init 0 */

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 9600;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */
  /* USER CODE END USART2_Init 2 */

}

/**
  * Enable DMA controller clock
  */
static void MX_DMA_Init(void)
{

  /* DMA controller clock enable */
  __HAL_RCC_DMA1_CLK_ENABLE();

  /* DMA interrupt init */
  /* DMA1_Channel1_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Channel1_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA1_Channel1_IRQn);

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
```

```
__HAL_RCC_GPIOC_CLK_ENABLE();
__HAL_RCC_GPIOF_CLK_ENABLE();
__HAL_RCC_GPIOA_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOC, REVERSE_Pin|BLDC_ON_Pin|PRESS_CLUTCH_Pin|RELEASE_GAS_Pin
                  |PRESS_GAS_Pin, GPIO_PIN_RESET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOA, LD2_Pin|LED_LEFT_Pin|LED_RIGHT_Pin|PRESS_BRAKE_Pin
                  |RELEASE_CLUTCH_Pin, GPIO_PIN_RESET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(RELEASE_BRAKE_GPIO_Port, RELEASE_BRAKE_Pin, GPIO_PIN_RESET);

/*Configure GPIO pins : REVERSE_Pin BLDC_ON_Pin PRESS_CLUTCH_Pin RELEASE_GAS_Pin
                   PRESS_GAS_Pin */
GPIO_InitStruct.Pin = REVERSE_Pin|BLDC_ON_Pin|PRESS_CLUTCH_Pin|RELEASE_GAS_Pin
                  |PRESS_GAS_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

/*Configure GPIO pins : BUTON_OFF_Pin SENSOR_RELEASE_BRAKE_Pin SENSOR_PRESS_BRAKE_Pin */
GPIO_InitStruct.Pin = BUTON_OFF_Pin|SENSOR_RELEASE_BRAKE_Pin|SENSOR_PRESS_BRAKE_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_PULLUP;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

/*Configure GPIO pins : LD2_Pin LED_LEFT_Pin LED_RIGHT_Pin PRESS_BRAKE_Pin
                   RELEASE_CLUTCH_Pin */
GPIO_InitStruct.Pin = LD2_Pin|LED_LEFT_Pin|LED_RIGHT_Pin|PRESS_BRAKE_Pin
                  |RELEASE_CLUTCH_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/*Configure GPIO pin : RELEASE_BRAKE_Pin */
GPIO_InitStruct.Pin = RELEASE_BRAKE_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(RELEASE_BRAKE_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pins : SENSOR_RELEASE_CLUTCH_Pin SENSOR_PRESS_CLUTCH_Pin */
GPIO_InitStruct.Pin = SENSOR_RELEASE_CLUTCH_Pin|SENSOR_PRESS_CLUTCH_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_PULLUP;
```

```
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

void braking(void)
{

  HAL_GPIO_WritePin(BLDC_ON_GPIO_Port, BLDC_ON_Pin, GPIO_PIN_RESET);
  set_acceleration(0);
  if(System_state.acc==0)
  {
   press_brake();
  }
  System_state.stop = 1;
}

void unbraking(void)
{
  HAL_GPIO_WritePin(BLDC_ON_GPIO_Port, BLDC_ON_Pin, GPIO_PIN_SET);
  release_bake();
  set_acceleration(25);
  System_state.stop = 0;
}

void transmition_set_N(void)
{
  System_state.direction = N;
}

void transmition_set_R(void)
{
  HAL_GPIO_WritePin(REVERSE_GPIO_Port, REVERSE_Pin, GPIO_PIN_RESET);
  System_state.direction = R;
}

void transmition_set_D(void)
{
  HAL_GPIO_WritePin(REVERSE_GPIO_Port, REVERSE_Pin, GPIO_PIN_SET);
  System_state.direction = D;
}

void start_motion(uint8_t direction)
{
  //stop pedal

unbraking();


      if((System_state.direction != direction) & System_state.acc > 0)
```

```
      {
       System_state.acc--;
       set_acceleration(System_state.acc);
       if(System_state.acc == 0)
       {
        direction_changed = true;
        LEFT_OFF();
        RIGHT_OFF();
        HAL_Delay(1000);
       }

      }
      else
      {
       if(direction == D)
        transmition_set_D();
       if(direction == R)
        transmition_set_R();

       if(direction_changed)
       {
        LEFT_OFF();
        RIGHT_OFF();
        HAL_Delay(1000);
        direction_changed = false;
       }
    //  if(System_state.clutch >= 1)
    //  {
    //   System_state.clutch--;
    //   set_clutch_position(System_state.clutch);
    //   HAL_Delay(3);
    //  }
       if((Global_state.accel > System_state.acc)&(Global_state.direction!=N))
       {
        System_state.acc++;
        set_acceleration(System_state.acc);
       }
       else
        if(System_state.acc > 0)
        {
         System_state.acc--;
         set_acceleration(System_state.acc);
        }
      }

}

void stop_motion(void)
{
```

```
     if(System_state.acc >=1)
     {
       System_state.acc--;
       set_acceleration(System_state.acc);

     }
     else
     {
       transmition_set_N();
       //stop pedal
     }

    if(System_state.acc <1)
    {
      braking();
    }
    else
    {
      System_state.acc--;
       set_acceleration(System_state.acc);
    }
}

void set_acceleration(uint8_t level)
{
  if(level < 25)
    level = 25;
  HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_1, DAC_ALIGN_12B_R, DAC_TRESHOLD + (DAC_MAX -
DAC_TRESHOLD)*level/100);
}

uint16_t percent;
int get_steer_angle(void)
{
//  if(System_state.steer_angle_adc > STEER_RIGHT)
//  {
    if(System_state.steer_angle_adc > STEER_AVERAGE)
    {
      percent = -(System_state.steer_angle_adc - STEER_AVERAGE)*100/(STEER_LEFT - STEER_AVERAGE);
      if(percent > 100)
        __NOP();
      return -(System_state.steer_angle_adc - STEER_AVERAGE)*100/(STEER_LEFT - STEER_AVERAGE);
    }
    else
    {
      percent = -(System_state.steer_angle_adc - STEER_AVERAGE)*100/(STEER_LEFT - STEER_AVERAGE);
      if(percent > 100)
        __NOP();
      return (System_state.steer_angle_adc - STEER_AVERAGE)*100/(STEER_RIGHT - STEER_AVERAGE);
    }
//  }
```

```
//  else
//    return System_state.steer_angle;

}

void set_steer_angle(void)
{
//#ifndef TEST

            if( Global_state.set_angle > 95 )
                        Global_state.set_angle = 95;
                        if( Global_state.set_angle < -95 )
                        Global_state.set_angle = -95;


  if( Global_state.set_angle > 0 ) //right
  {
   if( ((Global_state.set_angle - epsilon) <= System_state.steer_angle) & ((Global_state.set_angle +
epsilon) >= System_state.steer_angle) )
   {
    RIGHT_OFF(); //RIGHT
    LEFT_OFF();
   }
   else
   {
    if(Global_state.set_angle - epsilon > System_state.steer_angle)
    {
     RIGHT_ON(); //RIGHT
     LEFT_OFF();
    }
    else
     if(Global_state.set_angle + epsilon < System_state.steer_angle)
     {
      LEFT_ON(); //LEFT
      RIGHT_OFF();
     }
   }

  }
  else
   if( Global_state.set_angle < 0 ) //left
   {

    if( ((Global_state.set_angle - epsilon) <= System_state.steer_angle) & ((Global_state.set_angle +
epsilon) >= System_state.steer_angle) )
    {
     RIGHT_OFF(); //RIGHT
     LEFT_OFF();
    }
    else
    {
```

```
      if(Global_state.set_angle + epsilon < System_state.steer_angle)
      {
        LEFT_ON(); //LEFT
        RIGHT_OFF();
      }
      else
        if(Global_state.set_angle - epsilon > System_state.steer_angle)
        {
          RIGHT_ON(); //RIGHT
          LEFT_OFF();
        }
    }


  }
  else
  {
    if( (-1*epsilon <= System_state.steer_angle) & (System_state.steer_angle <= epsilon) )
    {
      RIGHT_OFF(); //RIGHT
      LEFT_OFF();
    }
    else
    {
      if(System_state.steer_angle > 0)
      {
        LEFT_ON(); //LEFT
        RIGHT_OFF();
      }
      else
        if(System_state.steer_angle < 0)
        {
          RIGHT_ON(); //RIGHT
          LEFT_OFF();
        }
    }

  }
//#endif
}

void turn_off_autopilot(void)
{
 RIGHT_OFF();
 LEFT_OFF();
 set_acceleration(0);
 unbraking();
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
```

```
 if(htim == &htim7)
 {
  //HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_6);
  Odometr_left_timer++;
  Odometr_right_timer++;
  acc_timer_left++;
  acc_timer_right++;
  led_adc_timer();

  if((Odometr_left_timer>=0xFFFE)||(Odometr_right_timer>=0xFFFE))
  {
   Odometr_left_timer=0;
   Odometr_right_timer=0;
  }

  counter_toggle++;
 // HAL_GPIO_TogglePin(TEST_GPIO_Port, TEST_Pin);
  if( counter_toggle>=500){
   TX_Uart_my.clutch_sensor_state=counter_com_tx;
   TX_UART_f();
   HAL_UART_Transmit_IT(&huart2, (uint8_t*)tx_buff, 47);
   counter_com_tx++;
    // HAL_GPIO_TogglePin(TEST_GPIO_Port, TEST_Pin);
   if(counter_com_tx==0xFF){counter_com_tx=0;}
   counter_toggle=0;
   //HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
 }
 // avr_potenciometr(steer_adc, 500);
  //filter_steer_angle=Filter_SMA(steer_adc);
 }
}
void led_adc_timer(void)
{
 if(Panel_struct.delay_left==DELAY_ACC_MAX)
    {
      Panel_struct.write_gpio_left=1;
    }
    else
    {
        if(acc_timer_left>=Panel_struct.delay_left)
        {
          if( HAL_GPIO_ReadPin(LED_LEFT_GPIO_Port,LED_LEFT_Pin)==1)
          {
             Panel_struct.write_gpio_left=0;
          }
          else
          {
             Panel_struct.write_gpio_left=1;
          }
          acc_timer_left=0;
```

```
      }
    }

 if(Panel_struct.delay_right==DELAY_ACC_MAX)
    {
      Panel_struct.write_gpio_right=1;
    }
    else
    {
     if(flag_off_tumbler==1){Panel_struct.write_gpio_right=0;}
     else{
     if(acc_timer_right>=Panel_struct.delay_right)
       {
         if( HAL_GPIO_ReadPin(LED_RIGHT_GPIO_Port,LED_RIGHT_Pin)==1)
         {
           Panel_struct.write_gpio_right=0;
         }
         else
         {
           Panel_struct.write_gpio_right=1;
         }
         acc_timer_right=0;
       }
     }
    }
}
void led_adc_set(uint8_t left_adc_gpio,uint8_t right_adc_gpio)
{
  if(left_adc_gpio==1){HAL_GPIO_WritePin(LED_LEFT_GPIO_Port,LED_LEFT_Pin, GPIO_PIN_SET);}
  else{HAL_GPIO_WritePin(LED_LEFT_GPIO_Port,LED_LEFT_Pin, GPIO_PIN_RESET);}

if(right_adc_gpio==1){HAL_GPIO_WritePin(LED_RIGHT_GPIO_Port,LED_RIGHT_Pin, GPIO_PIN_SET);}
  else{HAL_GPIO_WritePin(LED_RIGHT_GPIO_Port,LED_RIGHT_Pin, GPIO_PIN_RESET);}
}
uint16_t left_acc=0;
uint16_t right_acc=0;
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
 adc4Result = HAL_ADC_GetValue(&hadc1);
// if(System_state.acc!=0)
// {
 acc_left=HAL_ADC_GetValue(&hadc2);
 acc_right=HAL_ADC_GetValue(&hadc4);
 if(acc_left<=AK_ACC_LOW_LEFT){acc_left=AK_ACC_LOW_LEFT;}
 if(acc_right<=AK_ACC_LOW_RIGHT)
 {
   if(acc_right<=300){flag_off_tumbler=1;}
   else{flag_off_tumbler=0;}
   acc_right=AK_ACC_LOW_RIGHT;
 }
 left_acc=abs(acc_left-AK_ACC_LOW_LEFT)*100/AK_K_LEFT;
```

```
 right_acc=abs(acc_right-AK_ACC_LOW_RIGHT)*100/AK_K_RIGHT;
// }
  TX_Uart_my.gas_encoder_state[0]=left_acc&0x00FF;
  TX_Uart_my.gas_encoder_state[1]=(left_acc&0xFF00)>>8;
  TX_Uart_my.gas_encoder_state[2]= right_acc&0x00FF;
  TX_Uart_my.gas_encoder_state[3]= (right_acc&0xFF00)>>8;
}
/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */

  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/ /* USER CODE
BEGIN Header */
/**
  ******************************************************************************
  * @file          : main.c
  * @brief         : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
```

```
 * the "License"; You may not use this file except in compliance with the
 * License. You may obtain a copy of the License at:
 *                 opensource.org/licenses/BSD-3-Clause
 *
 ******************************************************************************
 */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "string.h"
#include "stdlib.h"
#include "stdbool.h"
#include "stdio.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

#define STEER_RIGHT     294//1374
#define STEER_AVERAGE   574//2237
#define STEER_LEFT      855//3257

#define DAC_MAX     4095

#define DAC_TRESHOLD    0
#define timeout        1000

#define RxBufSize 1024//1024
#define alpha2 0.95

#define SS_LR          45
#define SS_RR          65
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
#define LEFT_ON() TIM_CCxChannelCmd(htim3.Instance, TIM_CHANNEL_3, TIM_CCx_ENABLE) //RIGHT
#define LEFT_OFF() TIM_CCxChannelCmd(htim3.Instance, TIM_CHANNEL_3, TIM_CCx_DISABLE) //RIGHT
#define RIGHT_ON() TIM_CCxChannelCmd(htim2.Instance, TIM_CHANNEL_3, TIM_CCx_ENABLE) //LEFT
#define RIGHT_OFF() TIM_CCxChannelCmd(htim2.Instance, TIM_CHANNEL_3, TIM_CCx_DISABLE) //LEFT
#define BUTON_OFF()  HAL_GPIO_ReadPin(BUTON_OFF_GPIO_Port, BUTON_OFF_Pin)
/* USER CODE END PM */
```

```
/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;
ADC_HandleTypeDef hadc2;
ADC_HandleTypeDef hadc4;
DMA_HandleTypeDef hdma_adc1;

DAC_HandleTypeDef hdac1;

TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim2;
TIM_HandleTypeDef htim3;
TIM_HandleTypeDef htim7;

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */


typedef enum {
  N  = 0,
  D  = 1,
  R  = 2
} Direction_Type;

struct {
  bool autopilot_on;
  Direction_Type direction;
  int pressure;
  int steer_angle;
  int steer_angle_adc;
  bool motion;
  int manual_clutch;
  int clutch;
  bool stop;
  int acc;
  uint8_t stop_switch;
} System_state;

struct {
  int accel;
  int stop;
  int set_angle;
  Direction_Type direction;
  int Control;
}Global_state;

uint16_t acc_dac_level;
uint16_t steer_adc;

uint16_t acc_left;
uint16_t acc_right;
```

```
uint16_t acc_timer_left;
uint16_t acc_timer_right;

const char dir[3] = "NDR";
const char* autopilot[2] = {"OFF", "ON"};
const int epsilon = 10;

uint8_t receiveBuffer[RxBufSize];
char sendBuffer[RxBufSize];
char sendBuffer2[RxBufSize];

char *pars[12];
uint16_t tickstart;
bool command_received = false;
bool direction_changed = false;

uint32_t counter_toggle=0;
uint16_t adc4Result=0;
uint16_t procent_epwm=0;
uint32_t counter_adc=1;
uint32_t sum_adc=0;
int avr_steer_angle=0;
int filter_steer_angle=0;
int8_t test_dir=0;
uint8_t flag_off_tumbler=0;
uint8_t l=0;
uint8_t counter_com_tx=0;
struct {
  uint8_t Press_Brake_State;
  uint8_t Release_Brake_State;
  uint8_t Sensor_Press_Brake;
  uint8_t Sensor_Release_Brake;

  uint8_t Press_Gas_State;
  uint8_t Release_Gas_State;

  uint8_t Press_Clutch_State;
  uint8_t Release_Clutch_State;
  uint8_t Sensor_Press_Clutch;
  uint8_t Sensor_Release_Clutch;

  uint16_t Press_Brake_counter;
  uint16_t Release_Brake_counter;
  uint8_t Brake_Error;

  uint16_t Press_Gas_counter;
  uint16_t Release_Gas_counter;
  uint8_t Gas_Error;

  uint16_t Press_Clutch_counter;
  uint16_t Release_Clutch_counter;
```

```
  uint8_t Clutch_Error;
}Sensor_state={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

struct comand_pedals{
uint8_t comand_press_brake;
uint8_t comand_release_brake;
uint8_t comand_press_gas;
uint8_t comand_release_gas;
uint8_t comand_press_clutch;
uint8_t comand_release_clutch;
}Comand_pedals={0,0,0,0,0,0};

struct encoder_gas_state{
  uint32_t counter_raw;
  uint8_t direction;
}Encoder_gas_state={0,0};
uint8_t Buton_off=1;
uint16_t Odometr_left_timer=0;
uint16_t Odometr_right_timer=0;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_ADC1_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_DAC1_Init(void);
static void MX_TIM1_Init(void);
static void MX_TIM2_Init(void);
static void MX_TIM3_Init(void);
static void MX_TIM7_Init(void);
static void MX_ADC2_Init(void);
static void MX_ADC4_Init(void);
/* USER CODE BEGIN PFP */
void set_steer_angle(void);
int get_steer_angle(void);
void set_acceleration(uint8_t level);
void start_motion(uint8_t direction);
void stop_motion(void);
void turn_off_autopilot(void);
void error_brake(void);
void press_brake(void);
void release_bake(void);
void press_clutch(uint8_t press);
void led_adc_set(uint8_t left_adc_gpio,uint8_t right_adc_gpio);
void led_adc_timer(void);

extern void TX_UART_f(void);
uint8_t press_command=0;
extern void Odometr_function(uint16_t timer_left, uint16_t timer_right);
```

```
extern void Panel_functions(uint16_t timer_left, uint16_t timer_right, uint16_t acc_left, uint16_t
acc_right);
extern uint8_t buton_funciton(uint8_t gpio_state);
void braking(void);
/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
extern uint8_t flag_Timer7;
extern uint16_t Filter_SMA(uint16_t For_Filtered);
extern uint16_t Filter_Buffer[FILTER_SMA_ORDER];
extern uint8_t tx_buff[47];
extern tx_uart_my TX_Uart_my;
extern odometr_struct Odometr_left;
extern odometr_struct Odometr_right;
extern panel_struct Panel_struct;
uint8_t Pars_String(char **pars, char* str)
{
  char *pch;
  int y = 0;
  pch = strtok (str,"\",");
  while (pch != NULL)
    {
      pars[y++]=pch;
      pch = strtok (NULL, "\",");
    }
  return y;
}

void CheckUart()
{
  uint8_t *start_s_nump;
            uint8_t *start_s_nump_t;
  uint8_t start_s_num = 0;
#ifndef TEST
//  if(command_received)
//    if((HAL_GetTick() - tickstart) > timeout)
//      Global_state.stop = 1;
#endif

            /* P—P°C‰PëC,P° PsC, PïPµCЂPµPïPsP»PSPµPSPëC? P±CґC„PµCЂPsPI */
  if (huart2.RxXferCount <=0x80 )
                      {
                      /* PћC‡PëC?C,PєP° P±CґC„PµCЂPsPI */
     memset(receiveBuffer, 0, RxBufSize);
     HAL_UART_AbortReceive_IT(&huart2);
     HAL_UART_Receive_IT(&huart2, receiveBuffer, RxBufSize);
                      }

  /* P•C?P»Pë PSP°PNPrPµPS C?PëPjPIPsP» PєPsPSC†P° PïP°PєPµC,P° */
  if (strchr((char*)receiveBuffer, '#') != NULL )
```

```
  {

    /* P•C?P»Pë PSP°PNºPrPµPS C?PëPjPIPsP» PSP°C‡P°P»P° PïP°PєPµC,P° */
                        /* PïC‰PµPj PïPsC?P»PµPrPSPëPNº C?PëPjPIPsP» PSP°C‡P°P»P° PïP°PєPµC,P° PI
PïPsC?C‹P»PєPµ */
                        start_s_nump_t = receiveBuffer;
                        start_s_nump = NULL;
                        do
                                {
                                        start_s_nump_t = (uint8_t*)strchr((char*)start_s_nump_t, '*');
                                        if ( (start_s_nump_t !=NULL)  && (strchr((char*)start_s_nump_t,
'#' ) != NULL) )
                                                {
                                                        start_s_nump = start_s_nump_t;
                                                        start_s_nump_t++;
                                                }
                                } while (start_s_nump_t != NULL);

                        if ( start_s_nump == NULL)
                                return;
                    //  TX_UART_f();
                //   memset(sendBuffer2, 0, 128);
                        //memcpy(sendBuffer2, receiveBuffer, strlen(receiveBuffer));
                        //HAL_UART_Transmit_IT(&huart2, sendBuffer2, strlen(sendBuffer2));
                        //memset(sendBuffer2, 0, 128);
                        //memcpy(sendBuffer2, tx_buff, (strlen(tx_buff)+1));
                        //HAL_UART_Transmit_IT(&huart2, (uint8_t*)tx_buff, 42);


    if (start_s_nump != NULL)
     {
      start_s_num = start_s_nump - receiveBuffer;
      Pars_String(pars, (char*)start_s_nump + 1);
      //checkXOR();

      Global_state.accel = atoi(pars[0]);

      if (pars[1][0] == '1')
        Global_state.stop = 1;
      else
        Global_state.stop = 0;

      Global_state.set_angle = atoi(pars[2]);

     switch (pars[3][0])
      {
        case 'N':
          Global_state.direction = N;
          break;
        case 'D':
          {
```

```
            Global_state.direction = R;
           // TX_UART_f();
           // HAL_UART_Transmit_IT(&huart2, (uint8_t*)tx_buff, 42);
           break;
          }
        case 'R':
          Global_state.direction = D;
          break;
        default:
          Global_state.direction = N;
          break;
       }
       if ( !strcmp("ON", pars[4]) )
         Global_state.Control = 1;
       else
         Global_state.Control = 0;

       tickstart = HAL_GetTick();
       command_received = true;
      }

    /* PћC‡PëC?C,PєP° P±CґC„PµCЂPsPl */
    memset(receiveBuffer, 0, RxBufSize);
    HAL_UART_AbortReceive_IT(&huart2);
    HAL_UART_Receive_IT(&huart2, receiveBuffer, RxBufSize);
   }
}
void epvm_potenciometr(void)
{
//  HAL_ADC_Start(&hadc4);
                                                  //HAL_ADCEx_InjectedPollForConversion(&hadc4, 10);
 // adc4Result = HAL_ADC_GetValue(&hadc4);

  // TIM2->CCR3=(100*adc4Result)/4033;
  procent_epwm=TIM2->CCR3*100/100;
 // HAL_ADC_Stop(&hadc4);

   //HAL_Delay(300);
}
void avr_potenciometr(int row_angle, uint16_t max_counter_adc)
{
  counter_adc++;
    if( counter_adc>=max_counter_adc)
    {
      avr_steer_angle=sum_adc/counter_adc;
      counter_adc=0;
      sum_adc=row_angle;
    }
    else
    {
      sum_adc=sum_adc+row_angle;
```

```
   }
}
void Encoder_gas(void)
{
 Encoder_gas_state.counter_raw=TIM4->CNT;
 Encoder_gas_state.direction=((TIM4->CR1)&0x00000010)>1;
 /*TX_Uart_my.gas_encoder_state[0]=Encoder_gas_state.counter_raw&0x000F;
 TX_Uart_my.gas_encoder_state[1]=Encoder_gas_state.counter_raw&0x00F0>>8;
 TX_Uart_my.gas_encoder_state[2]=Encoder_gas_state.counter_raw&0x0F00>>16;
 TX_Uart_my.gas_encoder_state[3]=Encoder_gas_state.counter_raw&0xF000>>24; */
}
void Sensors_Condition (void)
{
//read comand gpio on Brake
 Sensor_state.Press_Brake_State=HAL_GPIO_ReadPin(GPIOA,PRESS_BRAKE_Pin);
 Sensor_state.Release_Brake_State=HAL_GPIO_ReadPin(GPIOB,RELEASE_BRAKE_Pin);
//read sensror gpio on Brake
 Sensor_state.Sensor_Press_Brake=HAL_GPIO_ReadPin(GPIOC,SENSOR_PRESS_BRAKE_Pin);
 Sensor_state.Sensor_Release_Brake=HAL_GPIO_ReadPin(GPIOC,SENSOR_RELEASE_BRAKE_Pin);
//read comand gpio on Clutch
 Sensor_state.Press_Clutch_State=HAL_GPIO_ReadPin(GPIOC,PRESS_CLUTCH_Pin);
 Sensor_state.Release_Clutch_State=HAL_GPIO_ReadPin(GPIOA,RELEASE_CLUTCH_Pin);

 //read sensor gpio on Clutch
 Sensor_state.Sensor_Press_Clutch=HAL_GPIO_ReadPin(GPIOA,SENSOR_PRESS_CLUTCH_Pin);
 Odometr_right.new_state_sensor=Sensor_state.Sensor_Press_Clutch;

 Sensor_state.Sensor_Release_Clutch=HAL_GPIO_ReadPin(GPIOA,SENSOR_RELEASE_CLUTCH_Pin);
 Odometr_left.new_state_sensor=Sensor_state.Sensor_Release_Clutch;

//read comand gpio on Gas
 Sensor_state.Press_Gas_State=HAL_GPIO_ReadPin(GPIOC,PRESS_GAS_Pin);
 Sensor_state.Release_Gas_State=HAL_GPIO_ReadPin(GPIOC,RELEASE_GAS_Pin);

 TX_Uart_my.brake_sensor_state=Sensor_state.Sensor_Press_Brake*10+
Sensor_state.Sensor_Release_Brake;
 // TX_Uart_my.clutch_sensor_state=Sensor_state.Sensor_Press_Clutch*10+
Sensor_state.Sensor_Release_Clutch;

 led_adc_set(Panel_struct.write_gpio_left,Panel_struct.write_gpio_right);
 //Encoder_gas();
}
void error_brake(void)
{
 if((Sensor_state.Press_Brake_State==1)&(Sensor_state.Release_Brake_State==1))
 {

   HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
   HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
   Sensor_state.Brake_Error=1;
   for(uint8_t i=0; i>=250;i++){}
```

```
  }
  else
  {
   if((Sensor_state.Sensor_Press_Brake==0)&(Sensor_state.Sensor_Release_Brake==0))
    {
     HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
     HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
     Sensor_state.Brake_Error=2;
     TX_Uart_my.brake_state=2;
     for(uint8_t i=0; i>=250;i++){}
    }
    else{
     if(Sensor_state.Sensor_Press_Brake==0){TX_Uart_my.brake_state=1;}
     if(Sensor_state.Sensor_Release_Brake==0){TX_Uart_my.brake_state=0;}
     Sensor_state.Brake_Error=0;}
  }
if((Sensor_state.Press_Clutch_State==1)&(Sensor_state.Release_Clutch_State==1))
  {
   HAL_GPIO_WritePin(GPIOC,PRESS_CLUTCH_Pin,GPIO_PIN_RESET);
   HAL_GPIO_WritePin(GPIOA,RELEASE_CLUTCH_Pin,GPIO_PIN_RESET);
   Sensor_state.Brake_Error=1;
   for(uint8_t i=0; i>=250;i++){}
  }
  else
  {
   if((Sensor_state.Sensor_Press_Clutch==0)&(Sensor_state.Sensor_Release_Clutch==0))
    {
     HAL_GPIO_WritePin(GPIOC,PRESS_CLUTCH_Pin,GPIO_PIN_RESET);
     HAL_GPIO_WritePin(GPIOA,RELEASE_CLUTCH_Pin,GPIO_PIN_RESET);
     Sensor_state.Clutch_Error=2;
     for(uint8_t i=0; i>=250;i++){}
    }
    else{Sensor_state.Clutch_Error=0;}
  }
}

void press_brake(void){

    if(Sensor_state.Sensor_Press_Brake==0)
    {
     HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
     HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
     for(uint8_t i=0; i>=250;i++){}
    }//sensor lights on
    else
    {

     if(Sensor_state.Brake_Error==0)
     {
       HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_SET);
       HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
```

```
        for(uint8_t i=0; i>=250;i++){}
      }
      else
      {
       HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
       HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
      }
    }

}
void release_bake(void){

   if(Sensor_state.Sensor_Release_Brake==0)
   {
     HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
     HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
     for(uint8_t i=0; i>=250;i++){}
   }//sensor lights on
   else
   {
    if(Sensor_state.Brake_Error==0)
    {
     HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
     HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_SET);
     for(uint8_t i=0; i>=250;i++){}
    }
    else
    {
      HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
      HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
    }
   }

}
void press_clutch(uint8_t press)
{
  switch (press)
   {
   case 0:
    {

     HAL_GPIO_WritePin(GPIOC,PRESS_CLUTCH_Pin,GPIO_PIN_RESET);
     HAL_GPIO_WritePin(GPIOA,RELEASE_CLUTCH_Pin,GPIO_PIN_RESET);
     break;
    }
   case 1:
    {

     if(Sensor_state.Sensor_Press_Clutch==1)
     {
      HAL_GPIO_WritePin(GPIOC,PRESS_CLUTCH_Pin,GPIO_PIN_SET);
```

```
        HAL_GPIO_WritePin(GPIOA,RELEASE_CLUTCH_Pin,GPIO_PIN_RESET);
       }
       else
       {
        HAL_GPIO_WritePin(GPIOC,PRESS_CLUTCH_Pin,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,RELEASE_CLUTCH_Pin,GPIO_PIN_RESET);
       }
       break;
     }
    case 2:
     {
       if(Sensor_state.Sensor_Release_Clutch==1)
       {
        HAL_GPIO_WritePin(GPIOC,PRESS_CLUTCH_Pin,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,RELEASE_CLUTCH_Pin,GPIO_PIN_SET);
       }
       else
       {
        HAL_GPIO_WritePin(GPIOC,PRESS_CLUTCH_Pin,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,RELEASE_CLUTCH_Pin,GPIO_PIN_RESET);
       }
       break;
     }
    }
}

void Comands_Pedals_Condition(void)
{
//  Brake


HAL_GPIO_WritePin(GPIOA,PRESS_BRAKE_Pin,GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB,RELEASE_BRAKE_Pin,GPIO_PIN_RESET);
    Sensor_state.Brake_Error=10;
//Clutch
}
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

/*Button_state.last_state=1;
Button_state.new_state=1;
Button_state.counter_flag_state=0;
Button_state.flag_Start_PWM=0;*/
  /* USER CODE END 1 */
```

```
 /* MCU Configuration--------------------------------------------------------*/

 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
 HAL_Init();

 /* USER CODE BEGIN Init */

 /* USER CODE END Init */

 /* Configure the system clock */
 SystemClock_Config();

 /* USER CODE BEGIN SysInit */

 /* USER CODE END SysInit */

 /* Initialize all configured peripherals */
 MX_GPIO_Init();
 MX_DMA_Init();
 MX_ADC1_Init();
 MX_USART2_UART_Init();
 MX_DAC1_Init();
 MX_TIM1_Init();
 MX_TIM2_Init();
 MX_TIM3_Init();
 MX_TIM7_Init();
 MX_ADC2_Init();
 MX_ADC4_Init();
 /* USER CODE BEGIN 2 */
 HAL_UART_Receive_IT(&huart2,          receiveBuffer, RxBufSize);

 //HAL_TIM_Base_Start(&htim1);
 HAL_TIM_Base_Start(&htim2);
 HAL_TIM_Base_Start(&htim3);



 HAL_DAC_Start(&hdac1, DAC_CHANNEL_1);

 set_acceleration(25);

//  set_acceleration(70);
//
//  set_acceleration(100);
 //HAL_TIM_PWM_Start (&htim1, TIM_CHANNEL_1);
HAL_TIM_PWM_Start (&htim3, TIM_CHANNEL_3);
 //HAL_TIM_PWM_Start_IT (&htim3, TIM_CHANNEL_3);
//HAL_TIM_Encoder_Start (&htim4, TIM_CHANNEL_1|TIM_CHANNEL_2);
HAL_TIM_Base_Start_IT(&htim7);
 HAL_ADC_Start_DMA(&hadc1, (uint32_t *)&steer_adc, 1);
```

```
  HAL_ADC_Start(&hadc1);
  //HAL_ADC_Start_DMA(&hadc2, (uint32_t *)&acc_left, 1);
  HAL_ADC_Start(&hadc2);
  //HAL_ADC_Start_DMA(&hadc4, (uint32_t *)&acc_right, 1);
  HAL_ADC_Start(&hadc4);
//  TIM1->CCR1=SS_LR;
  TIM2->CCR3=SS_LR;
  TIM3->CCR3=SS_RR;//right

LEFT_OFF();
RIGHT_OFF();
 memset(receiveBuffer, 0, 1024);
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
            System_state.steer_angle_adc= avr_steer_angle;
            System_state.steer_angle_adc = (uint16_t)((float)System_state.steer_angle_adc*alpha2 +
(float)steer_adc*(1-alpha2));
            System_state.steer_angle = get_steer_angle();
            CheckUart();
            Buton_off=buton_funciton(BUTON_OFF());

             if(Buton_off==0)
             {
               if(Global_state.Control)
                {
                                                HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
                    //HAL_Delay(300);
                    if(Global_state.stop == 1)
                    {
                      stop_motion();
                      if(Global_state.set_angle != System_state.steer_angle){set_steer_angle();}
                    }
                    else
                    {
                      start_motion(Global_state.direction);
                      if(Global_state.set_angle != System_state.steer_angle){set_steer_angle();}
                    }
                 }
                 else
                 {
                   HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
                   turn_off_autopilot();
                 }
             }
             else{
              //Buton_off=1;
```

```
            // System_state.acc==0;
             stop_motion();
             RIGHT_OFF(); //RIGHT
             LEFT_OFF();
            }

            error_brake();
            filter_steer_angle=Filter_SMA(steer_adc);
            avr_steer_angle= filter_steer_angle;
            Sensors_Condition();
            Panel_functions(acc_timer_left, acc_timer_right, acc_left, acc_right);

            TX_Uart_my.motion_state=Global_state.Control;
            TX_Uart_my.DAC_electric_motor[0]=System_state.acc;

          //
if(System_state.acc==0){Odometr_left.angular_velociy[0]=0;Odometr_right.angular_velociy[0]=0;}
          //  else
           /* {
                //if(test_dir>0)
                if(HAL_GPIO_ReadPin(REVERSE_GPIO_Port, REVERSE_Pin)>0)
                {
                   Odometr_left.new_state_direciton=1;
                   Odometr_right.new_state_direciton=1;
                }
                else
                {
                   Odometr_left.new_state_direciton=2;
                   Odometr_right.new_state_direciton=2;
                }*/
                                          Odometr_function(Odometr_left_timer,
Odometr_right_timer);
           // }
            TX_Uart_my.odometr_left[0]=Odometr_left.angular_velociy[0]&0x000000FF;
            TX_Uart_my.odometr_left[1]=(Odometr_left.angular_velociy[0]&0x0000FF00)>>8;
            TX_Uart_my.odometr_left[2]=(Odometr_left.angular_velociy[0]&0x00FF0000)>>16;
            TX_Uart_my.odometr_left[3]=(Odometr_left.angular_velociy[0]&0xFF000000)>>24;

            TX_Uart_my.odometr_right[0]=Odometr_right.angular_velociy[0]&0x000000FF;
            TX_Uart_my.odometr_right[1]=(Odometr_right.angular_velociy[0]&0x0000FF00)>>8;
            TX_Uart_my.odometr_right[2]=(Odometr_right.angular_velociy[0]&0x00FF0000)>>16;
            TX_Uart_my.odometr_right[3]=(Odometr_right.angular_velociy[0]&0xFF000000)>>24;

            TX_Uart_my.DAC_electric_motor[1]=0;
            TX_Uart_my.DAC_electric_motor[2]=0;

            TX_Uart_my.U_wheel_angle[0]=filter_steer_angle&0x00FF;
            TX_Uart_my.U_wheel_angle[1]=(filter_steer_angle&0xFF00)>>8;
            TX_Uart_my.U_wheel_angle[2]=0;
            TX_Uart_my.U_wheel_angle[3]=0;
```

```
            TX_Uart_my.wheel_angle[0]= System_state.steer_angle;
            TX_Uart_my.wheel_angle[1]=0;
            TX_Uart_my.wheel_angle[2]=0;
            TX_Uart_my.wheel_angle[3]=0;


                switch(System_state.direction)
               {
                case N:
                {
                  TX_Uart_my.control_condition[0]='N';
                  break;
                }
                case R:
                {
                  TX_Uart_my.control_condition[0]='D';
                  break;
                }case D:
                {
                  TX_Uart_my.control_condition[0]='R';
                  break;
                }
               }


//if(System_state.acc==0){Odometr_left.angular_velociy[0]=0;Odometr_right.angular_velociy[0]=0;}
/*if(test_dir==0){}
else
{
    //if(test_dir>0)
                                                 if(HAL_GPIO_ReadPin(REVERSE_GPIO_Port, REVERSE_Pin)>0)
    {
       Odometr_left.new_state_direciton=1;
       Odometr_right.new_state_direciton=1;
    }
    else
    {

       Odometr_left.new_state_direciton=2;
       Odometr_right.new_state_direciton=2;
    }

}*/
Odometr_function(Odometr_left_timer, Odometr_right_timer);
// l++;
//  if(Odometr_left.angular_velociy[0]==0){Odometr_left.angular_velociy[0]=0;}
// if(Odometr_right.angular_velociy[0]==0){Odometr_right.angular_velociy[0]=112+l;}
// if(l>=255){l=0;}

TX_Uart_my.odometr_left[0]=Odometr_left.angular_velociy[0]&0x000000FF;
TX_Uart_my.odometr_left[1]=(Odometr_left.angular_velociy[0]&0x0000FF00)>>8;
```

```
  TX_Uart_my.odometr_left[2]=(Odometr_left.angular_velociy[0]&0x00FF0000)>>16;
  TX_Uart_my.odometr_left[3]=(Odometr_left.angular_velociy[0]&0xFF000000)>>24;

  TX_Uart_my.odometr_right[0]=Odometr_right.angular_velociy[0]&0x000000FF;
  TX_Uart_my.odometr_right[1]=(Odometr_right.angular_velociy[0]&0x0000FF00)>>8;
  TX_Uart_my.odometr_right[2]=(Odometr_right.angular_velociy[0]&0x00FF0000)>>16;
  TX_Uart_my.odometr_right[3]=(Odometr_right.angular_velociy[0]&0xFF000000)>>24;

  TX_Uart_my.DAC_electric_motor[1]=0;
  TX_Uart_my.DAC_electric_motor[2]=0;

  TX_Uart_my.U_wheel_angle[0]=filter_steer_angle&0x00FF;
    TX_Uart_my.U_wheel_angle[1]=(filter_steer_angle&0xFF00)>>8;
     TX_Uart_my.U_wheel_angle[2]=0;
      TX_Uart_my.U_wheel_angle[3]=0;

  TX_Uart_my.wheel_angle[0]= System_state.steer_angle;
   TX_Uart_my.wheel_angle[1]=0;
    TX_Uart_my.wheel_angle[2]=0;
     TX_Uart_my.wheel_angle[3]=0;


      switch(System_state.direction)
     {
     case N:
      {
        TX_Uart_my.control_condition[0]='N';
        break;
      }
      case R:
      {
        TX_Uart_my.control_condition[0]='D';
        break;
      }case D:
      {
        TX_Uart_my.control_condition[0]='R';
        break;
      }
     }


  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
```

```
 RCC_OscInitTypeDef RCC_OscInitStruct = {0};
 RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
 RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};

 /** Initializes the RCC Oscillators according to the specified parameters
 * in the RCC_OscInitTypeDef structure.
 */
 RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
 RCC_OscInitStruct.HSIState = RCC_HSI_ON;
 RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
 RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
 RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
 RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
 RCC_OscInitStruct.PLL.PREDIV = RCC_PREDIV_DIV1;
 if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
 {
   Error_Handler();
 }
 /** Initializes the CPU, AHB and APB buses clocks
 */
 RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                       |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
 RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
 RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
 RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
 RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

 if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
 {
   Error_Handler();
 }
 PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART2|RCC_PERIPHCLK_TIM1
                       |RCC_PERIPHCLK_ADC12|RCC_PERIPHCLK_ADC34
                       |RCC_PERIPHCLK_TIM2|RCC_PERIPHCLK_TIM34;
 PeriphClkInit.Usart2ClockSelection = RCC_USART2CLKSOURCE_PCLK1;
 PeriphClkInit.Adc12ClockSelection = RCC_ADC12PLLCLK_DIV6;
 PeriphClkInit.Adc34ClockSelection = RCC_ADC34PLLCLK_DIV6;
 PeriphClkInit.Tim1ClockSelection = RCC_TIM1CLK_HCLK;
 PeriphClkInit.Tim2ClockSelection = RCC_TIM2CLK_HCLK;
 PeriphClkInit.Tim34ClockSelection = RCC_TIM34CLK_HCLK;
 if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
 {
   Error_Handler();
 }
}

/**
 * @brief ADC1 Initialization Function
 * @param None
 * @retval None
 */
```

```
static void MX_ADC1_Init(void)
{

  /* USER CODE BEGIN ADC1_Init 0 */

  /* USER CODE END ADC1_Init 0 */

  ADC_MultiModeTypeDef multimode = {0};
  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC1_Init 1 */

  /* USER CODE END ADC1_Init 1 */
  /** Common config
  */
  hadc1.Instance = ADC1;
  hadc1.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
  hadc1.Init.Resolution = ADC_RESOLUTION_10B;
  hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc1.Init.ContinuousConvMode = ENABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_RISING;
  hadc1.Init.ExternalTrigConv = ADC_EXTERNALTRIGCONV_T3_TRGO;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  hadc1.Init.DMAContinuousRequests = ENABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  hadc1.Init.LowPowerAutoWait = DISABLE;
  hadc1.Init.Overrun = ADC_OVR_DATA_OVERWRITTEN;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure the ADC multi-mode
  */
  multimode.Mode = ADC_MODE_INDEPENDENT;
  if (HAL_ADCEx_MultiModeConfigChannel(&hadc1, &multimode) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure Regular Channel
  */
  sConfig.Channel = ADC_CHANNEL_2;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SingleDiff = ADC_SINGLE_ENDED;
  sConfig.SamplingTime = ADC_SAMPLETIME_181CYCLES_5;
  sConfig.OffsetNumber = ADC_OFFSET_NONE;
  sConfig.Offset = 0;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
```

```
  }
  /* USER CODE BEGIN ADC1_Init 2 */

  /* USER CODE END ADC1_Init 2 */

}

/**
  * @brief ADC2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC2_Init(void)
{

  /* USER CODE BEGIN ADC2_Init 0 */

  /* USER CODE END ADC2_Init 0 */

  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC2_Init 1 */

  /* USER CODE END ADC2_Init 1 */
  /** Common config
  */
  hadc2.Instance = ADC2;
  hadc2.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
  hadc2.Init.Resolution = ADC_RESOLUTION_12B;
  hadc2.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc2.Init.ContinuousConvMode =  ENABLE;
  hadc2.Init.DiscontinuousConvMode = DISABLE;
  hadc2.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_RISING;
  hadc2.Init.ExternalTrigConv = ADC_EXTERNALTRIGCONV_T3_TRGO;
  hadc2.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc2.Init.NbrOfConversion = 1;
  hadc2.Init.DMAContinuousRequests = DISABLE;
  hadc2.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  hadc2.Init.LowPowerAutoWait = DISABLE;
  hadc2.Init.Overrun = ADC_OVR_DATA_OVERWRITTEN;
  if (HAL_ADC_Init(&hadc2) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure Regular Channel
  */
  sConfig.Channel = ADC_CHANNEL_8;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SingleDiff = ADC_SINGLE_ENDED;
  sConfig.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
  sConfig.OffsetNumber = ADC_OFFSET_NONE;
```

```
  sConfig.Offset = 0;
  if (HAL_ADC_ConfigChannel(&hadc2, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN ADC2_Init 2 */

  /* USER CODE END ADC2_Init 2 */

}

/**
  * @brief ADC4 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC4_Init(void)
{

  /* USER CODE BEGIN ADC4_Init 0 */

  /* USER CODE END ADC4_Init 0 */

  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC4_Init 1 */

  /* USER CODE END ADC4_Init 1 */
  /** Common config
  */
  hadc4.Instance = ADC4;
  hadc4.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
  hadc4.Init.Resolution = ADC_RESOLUTION_12B;
  hadc4.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc4.Init.ContinuousConvMode = ENABLE;
  hadc4.Init.DiscontinuousConvMode = DISABLE;
  hadc4.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc4.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc4.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc4.Init.NbrOfConversion = 1;
  hadc4.Init.DMAContinuousRequests = DISABLE;
  hadc4.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  hadc4.Init.LowPowerAutoWait = DISABLE;
  hadc4.Init.Overrun = ADC_OVR_DATA_OVERWRITTEN;
  if (HAL_ADC_Init(&hadc4) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure Regular Channel
  */
  sConfig.Channel = ADC_CHANNEL_3;
```

```
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SingleDiff = ADC_SINGLE_ENDED;
  sConfig.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
  sConfig.OffsetNumber = ADC_OFFSET_NONE;
  sConfig.Offset = 0;
  if (HAL_ADC_ConfigChannel(&hadc4, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN ADC4_Init 2 */

  /* USER CODE END ADC4_Init 2 */

}

/**
  * @brief DAC1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_DAC1_Init(void)
{

  /* USER CODE BEGIN DAC1_Init 0 */

  /* USER CODE END DAC1_Init 0 */

  DAC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN DAC1_Init 1 */

  /* USER CODE END DAC1_Init 1 */
  /** DAC Initialization
  */
  hdac1.Instance = DAC1;
  if (HAL_DAC_Init(&hdac1) != HAL_OK)
  {
    Error_Handler();
  }
  /** DAC channel OUT1 config
  */
  sConfig.DAC_Trigger = DAC_TRIGGER_NONE;
  sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;
  if (HAL_DAC_ConfigChannel(&hdac1, &sConfig, DAC_CHANNEL_1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN DAC1_Init 2 */

  /* USER CODE END DAC1_Init 2 */
```

```
}

/**
  * @brief TIM1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM1_Init(void)
{

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 720-1;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 100-1;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterOutputTrigger2 = TIM_TRGO2_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */

  /* USER CODE END TIM1_Init 2 */

}

/**
  * @brief TIM2 Initialization Function
```

```
  * @param None
  * @retval None
  */
static void MX_TIM2_Init(void)
{

  /* USER CODE BEGIN TIM2_Init 0 */

  /* USER CODE END TIM2_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM2_Init 1 */

  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 720-1;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 100-1;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 26;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_3) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */
```

```
  /* USER CODE END TIM2_Init 2 */
  HAL_TIM_MspPostInit(&htim2);

}

/**
  * @brief TIM3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM3_Init(void)
{

  /* USER CODE BEGIN TIM3_Init 0 */

  /* USER CODE END TIM3_Init 0 */

  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM3_Init 1 */

  /* USER CODE END TIM3_Init 1 */
  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 720-1;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim3.Init.Period = 100-1;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_PWM_Init(&htim3) !=  HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_3) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */
```

```
  HAL_TIM_MspPostInit(&htim3);

}

/**
  * @brief TIM7 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM7_Init(void)
{

  /* USER CODE BEGIN TIM7_Init 0 */

  /* USER CODE END TIM7_Init 0 */

  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM7_Init 1 */

  /* USER CODE END TIM7_Init 1 */
  htim7.Instance = TIM7;
  htim7.Init.Prescaler = 1800;
  htim7.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim7.Init.Period = 100;
  htim7.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim7) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim7, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM7_Init 2 */

  /* USER CODE END TIM7_Init 2 */

}

/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{

  /* USER CODE BEGIN USART2_Init 0 */
```

```
  /* USER CODE END USART2_Init 0 */

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 9600;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */
  /* USER CODE END USART2_Init 2 */

}

/**
  * Enable DMA controller clock
  */
static void MX_DMA_Init(void)
{

  /* DMA controller clock enable */
  __HAL_RCC_DMA1_CLK_ENABLE();

  /* DMA interrupt init */
  /* DMA1_Channel1_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Channel1_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA1_Channel1_IRQn);

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
```

```
__HAL_RCC_GPIOC_CLK_ENABLE();
__HAL_RCC_GPIOF_CLK_ENABLE();
__HAL_RCC_GPIOA_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOC, REVERSE_Pin|BLDC_ON_Pin|PRESS_CLUTCH_Pin|RELEASE_GAS_Pin
                  |PRESS_GAS_Pin, GPIO_PIN_RESET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOA, LD2_Pin|LED_LEFT_Pin|LED_RIGHT_Pin|PRESS_BRAKE_Pin
                  |RELEASE_CLUTCH_Pin, GPIO_PIN_RESET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(RELEASE_BRAKE_GPIO_Port, RELEASE_BRAKE_Pin, GPIO_PIN_RESET);

/*Configure GPIO pins : REVERSE_Pin BLDC_ON_Pin PRESS_CLUTCH_Pin RELEASE_GAS_Pin
                   PRESS_GAS_Pin */
GPIO_InitStruct.Pin = REVERSE_Pin|BLDC_ON_Pin|PRESS_CLUTCH_Pin|RELEASE_GAS_Pin
                  |PRESS_GAS_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

/*Configure GPIO pins : BUTON_OFF_Pin SENSOR_RELEASE_BRAKE_Pin SENSOR_PRESS_BRAKE_Pin */
GPIO_InitStruct.Pin = BUTON_OFF_Pin|SENSOR_RELEASE_BRAKE_Pin|SENSOR_PRESS_BRAKE_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_PULLUP;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

/*Configure GPIO pins : LD2_Pin LED_LEFT_Pin LED_RIGHT_Pin PRESS_BRAKE_Pin
                   RELEASE_CLUTCH_Pin */
GPIO_InitStruct.Pin = LD2_Pin|LED_LEFT_Pin|LED_RIGHT_Pin|PRESS_BRAKE_Pin
                  |RELEASE_CLUTCH_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/*Configure GPIO pin : RELEASE_BRAKE_Pin */
GPIO_InitStruct.Pin = RELEASE_BRAKE_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(RELEASE_BRAKE_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pins : SENSOR_RELEASE_CLUTCH_Pin SENSOR_PRESS_CLUTCH_Pin */
GPIO_InitStruct.Pin = SENSOR_RELEASE_CLUTCH_Pin|SENSOR_PRESS_CLUTCH_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_PULLUP;
```

```
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

void braking(void)
{

  HAL_GPIO_WritePin(BLDC_ON_GPIO_Port, BLDC_ON_Pin, GPIO_PIN_RESET);
  set_acceleration(0);
  if(System_state.acc==0)
  {
   press_brake();
  }
  System_state.stop = 1;
}

void unbraking(void)
{
  HAL_GPIO_WritePin(BLDC_ON_GPIO_Port, BLDC_ON_Pin, GPIO_PIN_SET);
  release_bake();
  set_acceleration(25);
  System_state.stop = 0;
}

void transmition_set_N(void)
{
  System_state.direction = N;
}

void transmition_set_R(void)
{
  HAL_GPIO_WritePin(REVERSE_GPIO_Port, REVERSE_Pin, GPIO_PIN_RESET);
  System_state.direction = R;
}

void transmition_set_D(void)
{
  HAL_GPIO_WritePin(REVERSE_GPIO_Port, REVERSE_Pin, GPIO_PIN_SET);
  System_state.direction = D;
}

void start_motion(uint8_t direction)
{
  //stop pedal

unbraking();


      if((System_state.direction != direction) & System_state.acc > 0)
```

```
      {
       System_state.acc--;
       set_acceleration(System_state.acc);
       if(System_state.acc == 0)
       {
        direction_changed = true;
        LEFT_OFF();
        RIGHT_OFF();
        HAL_Delay(1000);
       }

      }
      else
      {
       if(direction == D)
        transmition_set_D();
       if(direction == R)
        transmition_set_R();

       if(direction_changed)
       {
        LEFT_OFF();
        RIGHT_OFF();
        HAL_Delay(1000);
        direction_changed = false;
       }
    //  if(System_state.clutch >= 1)
    //  {
    //   System_state.clutch--;
    //   set_clutch_position(System_state.clutch);
    //   HAL_Delay(3);
    //  }
       if((Global_state.accel > System_state.acc)&(Global_state.direction!=N))
       {
        System_state.acc++;
        set_acceleration(System_state.acc);
       }
       else
        if(System_state.acc > 0)
        {
         System_state.acc--;
         set_acceleration(System_state.acc);
        }
      }

}

void stop_motion(void)
{
```

```
     if(System_state.acc >=1)
     {
       System_state.acc--;
       set_acceleration(System_state.acc);

     }
     else
     {
       transmition_set_N();
       //stop pedal
     }

    if(System_state.acc <1)
    {
      braking();
    }
    else
    {
      System_state.acc--;
       set_acceleration(System_state.acc);
    }
}

void set_acceleration(uint8_t level)
{
  if(level < 25)
    level = 25;
  HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_1, DAC_ALIGN_12B_R, DAC_TRESHOLD + (DAC_MAX -
DAC_TRESHOLD)*level/100);
}

uint16_t percent;
int get_steer_angle(void)
{
//  if(System_state.steer_angle_adc > STEER_RIGHT)
//  {
    if(System_state.steer_angle_adc > STEER_AVERAGE)
    {
      percent = -(System_state.steer_angle_adc - STEER_AVERAGE)*100/(STEER_LEFT - STEER_AVERAGE);
      if(percent > 100)
        __NOP();
      return -(System_state.steer_angle_adc - STEER_AVERAGE)*100/(STEER_LEFT - STEER_AVERAGE);
    }
    else
    {
      percent = -(System_state.steer_angle_adc - STEER_AVERAGE)*100/(STEER_LEFT - STEER_AVERAGE);
      if(percent > 100)
        __NOP();
      return (System_state.steer_angle_adc - STEER_AVERAGE)*100/(STEER_RIGHT - STEER_AVERAGE);
    }
//  }
```

```
//  else
//    return System_state.steer_angle;

}

void set_steer_angle(void)
{
//#ifndef TEST

          if( Global_state.set_angle > 95 )
                    Global_state.set_angle = 95;
                    if( Global_state.set_angle < -95 )
                    Global_state.set_angle = -95;


  if( Global_state.set_angle > 0 ) //right
  {
   if( ((Global_state.set_angle - epsilon) <= System_state.steer_angle) & ((Global_state.set_angle +
epsilon) >= System_state.steer_angle) )
   {
    RIGHT_OFF(); //RIGHT
    LEFT_OFF();
   }
   else
   {
    if(Global_state.set_angle - epsilon > System_state.steer_angle)
    {
     RIGHT_ON(); //RIGHT
     LEFT_OFF();
    }
    else
     if(Global_state.set_angle + epsilon < System_state.steer_angle)
     {
      LEFT_ON(); //LEFT
      RIGHT_OFF();
     }
   }

  }
  else
   if( Global_state.set_angle < 0 ) //left
   {

    if( ((Global_state.set_angle - epsilon) <= System_state.steer_angle) & ((Global_state.set_angle +
epsilon) >= System_state.steer_angle) )
    {
     RIGHT_OFF(); //RIGHT
     LEFT_OFF();
    }
    else
    {
```

```
      if(Global_state.set_angle + epsilon < System_state.steer_angle)
      {
        LEFT_ON(); //LEFT
        RIGHT_OFF();
      }
      else
        if(Global_state.set_angle - epsilon > System_state.steer_angle)
        {
          RIGHT_ON(); //RIGHT
          LEFT_OFF();
        }
    }


  }
  else
  {
    if( (-1*epsilon <= System_state.steer_angle) & (System_state.steer_angle <= epsilon) )
    {
      RIGHT_OFF(); //RIGHT
      LEFT_OFF();
    }
    else
    {
      if(System_state.steer_angle > 0)
      {
        LEFT_ON(); //LEFT
        RIGHT_OFF();
      }
      else
        if(System_state.steer_angle < 0)
        {
          RIGHT_ON(); //RIGHT
          LEFT_OFF();
        }
    }

  }
//#endif
}

void turn_off_autopilot(void)
{
 RIGHT_OFF();
 LEFT_OFF();
 set_acceleration(0);
 unbraking();
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
```

```
 if(htim == &htim7)
 {
  //HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_6);
  Odometr_left_timer++;
  Odometr_right_timer++;
  acc_timer_left++;
  acc_timer_right++;
  led_adc_timer();

  if((Odometr_left_timer>=0xFFFE)||(Odometr_right_timer>=0xFFFE))
  {
   Odometr_left_timer=0;
   Odometr_right_timer=0;
  }

  counter_toggle++;
 // HAL_GPIO_TogglePin(TEST_GPIO_Port, TEST_Pin);
  if( counter_toggle>=500){
   TX_Uart_my.clutch_sensor_state=counter_com_tx;
   TX_UART_f();
   HAL_UART_Transmit_IT(&huart2, (uint8_t*)tx_buff, 47);
   counter_com_tx++;
    // HAL_GPIO_TogglePin(TEST_GPIO_Port, TEST_Pin);
   if(counter_com_tx==0xFF){counter_com_tx=0;}
   counter_toggle=0;
   //HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
 }
 // avr_potenciometr(steer_adc, 500);
  //filter_steer_angle=Filter_SMA(steer_adc);
 }
}
void led_adc_timer(void)
{
 if(Panel_struct.delay_left==DELAY_ACC_MAX)
    {
      Panel_struct.write_gpio_left=1;
    }
    else
    {
        if(acc_timer_left>=Panel_struct.delay_left)
        {
          if( HAL_GPIO_ReadPin(LED_LEFT_GPIO_Port,LED_LEFT_Pin)==1)
          {
             Panel_struct.write_gpio_left=0;
          }
          else
          {
             Panel_struct.write_gpio_left=1;
          }
          acc_timer_left=0;
```

```
      }
    }

 if(Panel_struct.delay_right==DELAY_ACC_MAX)
    {
      Panel_struct.write_gpio_right=1;
    }
    else
    {
     if(flag_off_tumbler==1){Panel_struct.write_gpio_right=0;}
     else{
     if(acc_timer_right>=Panel_struct.delay_right)
       {
         if( HAL_GPIO_ReadPin(LED_RIGHT_GPIO_Port,LED_RIGHT_Pin)==1)
         {
           Panel_struct.write_gpio_right=0;
         }
         else
         {
           Panel_struct.write_gpio_right=1;
         }
         acc_timer_right=0;
       }
     }
    }
}
void led_adc_set(uint8_t left_adc_gpio,uint8_t right_adc_gpio)
{
  if(left_adc_gpio==1){HAL_GPIO_WritePin(LED_LEFT_GPIO_Port,LED_LEFT_Pin, GPIO_PIN_SET);}
  else{HAL_GPIO_WritePin(LED_LEFT_GPIO_Port,LED_LEFT_Pin, GPIO_PIN_RESET);}

if(right_adc_gpio==1){HAL_GPIO_WritePin(LED_RIGHT_GPIO_Port,LED_RIGHT_Pin, GPIO_PIN_SET);}
  else{HAL_GPIO_WritePin(LED_RIGHT_GPIO_Port,LED_RIGHT_Pin, GPIO_PIN_RESET);}
}
uint16_t left_acc=0;
uint16_t right_acc=0;
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
 adc4Result = HAL_ADC_GetValue(&hadc1);
// if(System_state.acc!=0)
// {
 acc_left=HAL_ADC_GetValue(&hadc2);
 acc_right=HAL_ADC_GetValue(&hadc4);
 if(acc_left<=AK_ACC_LOW_LEFT){acc_left=AK_ACC_LOW_LEFT;}
 if(acc_right<=AK_ACC_LOW_RIGHT)
 {
   if(acc_right<=300){flag_off_tumbler=1;}
   else{flag_off_tumbler=0;}
   acc_right=AK_ACC_LOW_RIGHT;
 }
 left_acc=abs(acc_left-AK_ACC_LOW_LEFT)*100/AK_K_LEFT;
```

```
 right_acc=abs(acc_right-AK_ACC_LOW_RIGHT)*100/AK_K_RIGHT;
// }
  TX_Uart_my.gas_encoder_state[0]=left_acc&0x00FF;
  TX_Uart_my.gas_encoder_state[1]=(left_acc&0xFF00)>>8;
  TX_Uart_my.gas_encoder_state[2]= right_acc&0x00FF;
  TX_Uart_my.gas_encoder_state[3]= (right_acc&0xFF00)>>8;
}
/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */

  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```